Proceedings 2009

WAGENINGEN UNIVERSITY
WAGENINGEN UR

Proceedings of the 7th Field Robot Event 2009

Wageningen, July 6 & 7, 2009

ISBN	:	978-90-8585-744-0
Date	:	October 2009
Publisher	:	Wageningen University Farm Technology Group
Adress	:	Technotron (Building 118) Bornse Weilanden 9 6708 WG Wageningen The Netherlands Tel: +31 317 482980 Fax: +31 317 484819
Contact	:	info@fieldrobot.com

WAGENINGEN UNIVERSITEIT
WAGENINGEN UR

Sponsored by:

Amazone

AVR

Claas

DLG

Fachhochschule Osnabrück

Future Farm

Gemeente Wageningen

Kverneland Group

NVTL

Rauch

Schuitemaker

Unifarm

WUR Glastuinbouw

Wageningen UR

Albron

PREFACE

Idea Field Robot Event

7th Field Robot Event 2009 – how an idea takes off

It was in September 2002 when the story started: in a train back home from Budapest, where Wageningen Agrotechnology students won the vision award of the EurAgEng-Conference. That price-winning vision was about robots in agriculture. Why shouldn't this vision become more real and why shouldn't more students participate in such exciting activities – e.g. in a contest for small robots competing in a real field? The idea was born and preparations started immediately to invite students from all over Europe for the 1st Field Robot Event in June 2003.

Of course, it was not the first robot design contest the world has ever seen. Famous archetype is the student design competition of the Massachusetts Institute of Technology (MIT). Also popular versions of robot competitions on TV attract public attention. Anyhow, there are a lot of unique elements in the Field Robot Event:

- The robots are acting outdoors, in a harsh and unstructured environment with varying light and climatic conditions – a challenge even for professional research groups,
- The robots are not attacking each other, but competing in an Olympic manner to identify the best,
- The choice of components is not limited to a standard kit to allow unhampered creativity,
- The cost/performance relation of the robots is taken into consideration,
- The student teams present their creations on a conference-like robot fair and write a paper to be published in the Proceedings of the Field Robot Event.

In total, 16 Field Robot teams registered this year. Wageningen UR is looking forward to host guests from Germany, Finland, Denmark, Czech Republic, Slovenia and the Netherlands.

Parallel with the Field Robot event, supported by the EU FutureFarm project (www.futurefarm.eu), research groups from the United Kingdom, Denmark and the Netherlands will demonstrate research prototypes of professional agricultural robots.

Already now, I am thanking the sponsors for financial support as well as the students and staff members for the enthusiastic dedication during the preparatory activities and the event itself.

Also this week, Wageningen hosts the Joint International Agricultural Conference - JIAC 2009. Special thanks to the organizers of JIAC2009 for a very constructive collaboration. It is a pleasure and honour to present both robot activities to the scientific community attending this conference.

I wish all participants and visitors a pleasant 7th Field Robot Event – an untroubled festival amongst friends with plenty of inspiring impressions.

Prof. dr ir E.J. van Henten

Wageningen UR, Department of Agrotechnology and Food Sciences

Farm Technology Group

INDEX

Comeback of the modular agricultural sensor platform „Amaizeing“

Hennewig, Alexander; Kraatz, Franz; Numonov, Alisehr; Strangar, Boris; Wessel, Sven

University of Applied Sciences Osnabrück, Faculty of Engineering and Computer Science/ Interdisciplinary Research Center Intelligent Sensor Systems (ISYS), Albrechtstr. 30, 49076 Osnabrück/Germany, Phone +49541 969 2090

Abstract

The autonomous field robot Amaizeing'09 has been built by a group of five students of the University of Applied Sciences Osnabrück to participate at the Field Robot Event 2009 in Wageningen. It is based on the field robot Amaizeing that has already successfully participated in the Field Robot Event 2007. The robot's technology is based on microcontrollers combined with a sensor fusion concept using low-cost sensors with decentralized data processing. To fulfil the requirements of the Field Robot Event's tasks of the year 2009 the system was extended by a new camera technology, including color filters and new image processing routines, and new software algorithms for the autonomous navigation and turn. Moreover, new algorithms and actuators for the freestyle session of the Field Robot Event 2009, simulating the autonomous process of placing and watering of maize plants on the field in cooperation with the field robot "smartNAD", were developed.

Figure 1: Field robot Amaizeing'09

Keywords: Field Robot Event, autonomous robots, sensor fusion, modular sensor concept, maize

Introduction

The 7th Field Robot Event takes place at Wageningen UR from 6th to 8th of July 2009. It is the sixth time in a row for a student team of the University of Applied Sciences Osnabrück taking part in this

international competition with an own platform ([EyeMaize 2004, optoMAIZER 2005, Maizerati 2006, Amaizeing 2007, Agronaut 2008]). This year's team consist of five electrical engineering (Bachelor) students building the robot parallel to their studies. As a base for their robot called Amaizeing'09 the student team decided to use the already existing field robot Amaizeing that has finished the Field Robot Event 2007 in second place. This enabled the student to benefit of the technology and knowledge of previous developments of the teams of the University of Applied Sciences Osnabrück that has been combined in this successful field robot. To make the robot useable for the tasks of the Field Robot Event 2009 - including freestyle, navigation, advanced navigation and weed control - the student team had to extend the robot system by integrating new software and hardware strategies. Besides this it was necessary to get around the known problems of the Amaizeing that had shown up during its performance in the year 2007 [Amaizeing 2007]

The differences in the tasks compared to the ones of the Field Robot Event 2007 are:

Robust Navigation

This task is similar to the one of 2007. The robot has to drive through curved rows and to make a head-land turn at the end of the row into the next one.

Advanced Navigation

In the second task the robot has to drive along a predefined pattern within straight maize rows with missing plants. Arriving at the headland, the robot should drive to a predefined row. The difference to the task of 2007 are the rounded off ends of the maize rows (see figure 2).

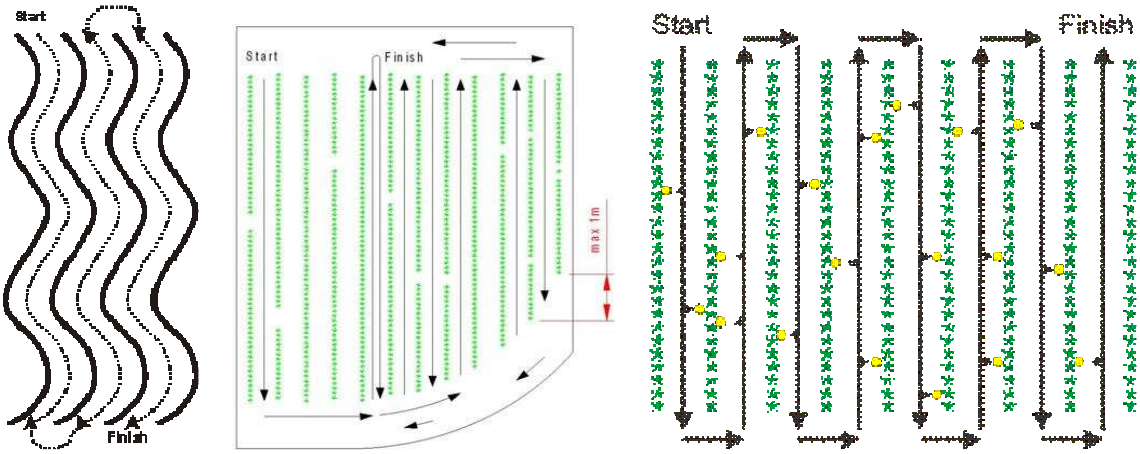

Figure 2: Tasks of the Field Robot Event 2009 (left: Robust Navigation, middle: Advanced Navigation, right: Weed control) [http://www.fieldrobot.nl]

'Weed' control

This task experienced a big change. In the last years the robot has to detect randomly placed yellow golf balls during the third task. This year the golf balls are green. In addition water sensitively strips are laid out for accuracy examination. When the robot detects this 'weed' it signalizes this by producing a flash-light or sound and performs a weed killing action

Enhancement of the field robot Amaizeing

The Amaizeing is a four wheel field robot based on a moster truck model. The system structure of the Amaizeing is based on a decentralized processing concept of sensor data [Amaizeing 2007] using a central CAN bus for data exchange. As a centre for the system structure a 16-Bit Infineon C167 microcontroller equipped with the RTXtiny real time operating system made by KEIL is used for

the calculation of the software algorithms for the main tasks like the autonomous navigation and turn. Besides this, another 16-Bit FUJITSU microcontroller is used for calculating the image analysis algorithms for the weed control task. The low cost sensors integrated in the system structure are triangulation IR distance sensors for the navigation, two smart cameras CMUcam2 for the weed detection and navigation, a gyroscope, an optical mouse sensor for gathering slip free odometry data and an additional hall sensor for distance measurements.

Load-sensitive engine regulation

The most important known problem of the Amaizeing was the high speed dependence of changing loads or inclines. This dependence had a bad influence on the quality and robustness of the navigation and turn algorithms because of their high speed dependence. This problem was a result of the missing speed control of the motors. Since the robot has to be able to carry heavy water tanks for the weed control actuator with changing weight resulting of the water sprayed during the task the robot system has to handle changing loads. To get around this problem it was necessary to integrate a speed control into the robot's system.

Since a reliable speed control depends on the quality of the speed measured by the sensor systems, it was decided to use the slip free and high resolution odometry data of the optical mouse sensor instead of using the integrated hall sensor. With the missing hardware to interface with the mouse sensor, this decision made it impossible to use common speed control modules available for the type motors used by the vehicle.

$$u(t) = K_p \cdot e(t) \qquad u(t) = K_p \left(e(t) + \frac{1}{T_N} \int_0^t e(t')dt' \right)$$

As a result the speed control was implemented as software algorithms. These algorithms include a P- and a PI-controller that can be parameterized by using the Amaizeing GUI [Amaizeing 2007]. The controllers were implemented as a special thread into the real time operating system of the C167 microcontroller (see figure 3).

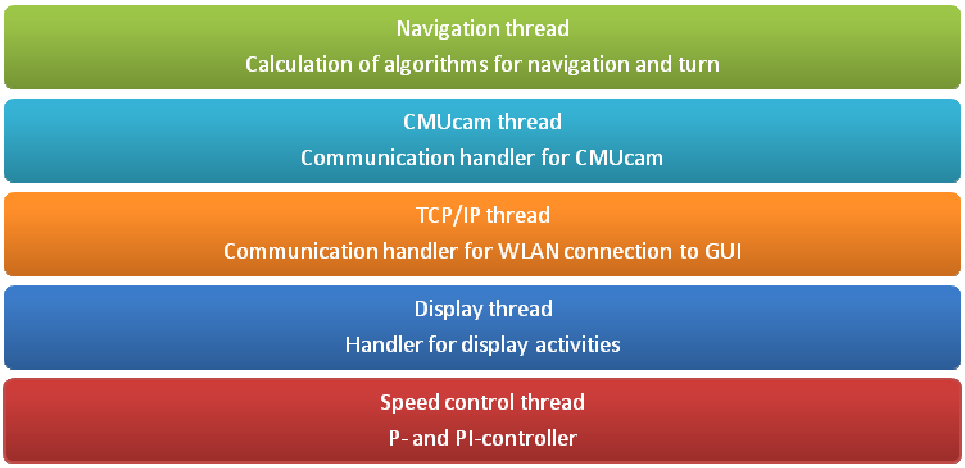

Figure 3: Threads of the real time operating system

Advances Navigation with rounded ends

The most important change to the advanced navigation task for Field Robot Event 2009 was the different length of consecutive maize rows. This leads to varying turning angles necessary for the autonomous head-land turn. Since, for the last Field Robot Events, it was sufficient to combine the turn process of sequential 90° turns and predefined distances to be driven (see figure 4 left side), it was necessary to implement a new strategy for the head-land turn. Because of the impossible ability

to create a dynamic map while driving through the maize rows, related to the used low-cost sensors, it was decided to integrate a virtual map of the predefined pattern into the robots software. Therefore the graphical user interface was enhanced to be able to parameterize and visualize the necessary row parameters and the row itself.

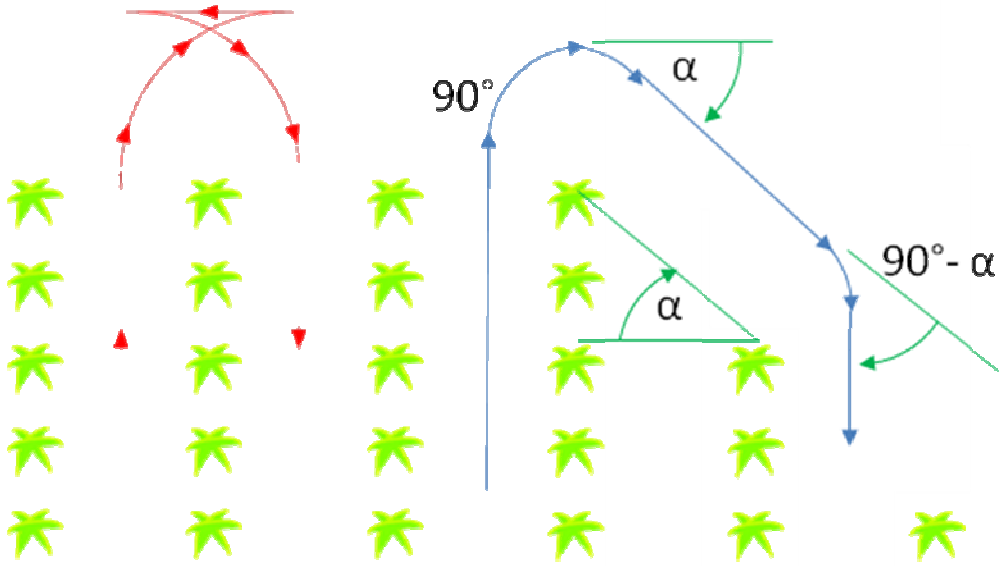

Figure 4: Implemented turn strategies

With this information given, the robot was able to calculate the necessary sequence of turn angles and distances (see figure 4 right side) for the autonomous headland turn. To perform this turn the integrated gyroscope is used to control the turning angle of the robot. The distance driven is measured by the optical mouse sensor. Besides the driven distance also the appearing rows to the side will be counted while driving past them.

Weed control with green golf balls

With the colour of the golf balls changed from yellow to green it has become more difficult to distinguish maize and the ball lying in between them. To still be able to successfully perform the weed detection with the help of a low-cost smart camera it was necessary to increase the contrast between the golf ball and its surroundings. The first step performed was a spectral analysis of the ball's colour to find wavelength within its colour spectrum (see figure 5) with a high intensity and to compare it to the ones of maize plants and soil.

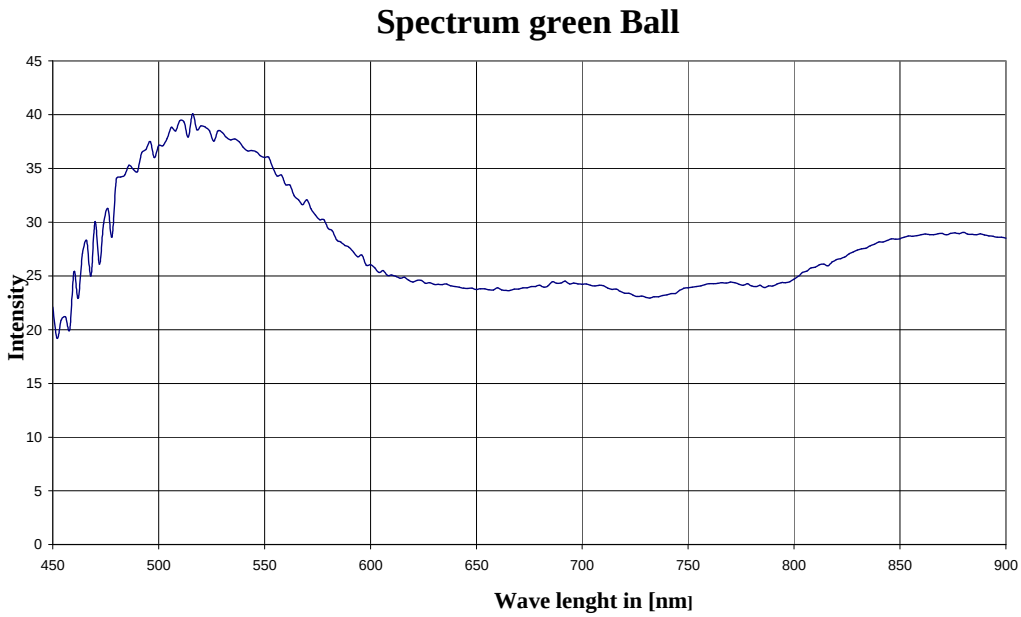

Figure 5: Reflectance spectrum of the green ball

The highest reflexion of the ball appeared in a range from 500 to 520 nm. To be able to check the contrast of ball compared to the one of maize an adjustable optical band pass filter was mounted to the smart camera. With this assembly it was possible to take pictures of the golf ball lying in the maize rows while varying the band pass' wavelength. After an analysis of the images the highest contrast was found at a wavelength of about 503 nm. With a band pass of 503 nm +/- 15 nm the surroundings of the ball were almost invisible for the camera. As a result of this analysis the camera's lens was equipped with an optical band pass filter of the mentioned wavelength range (see figure 6).

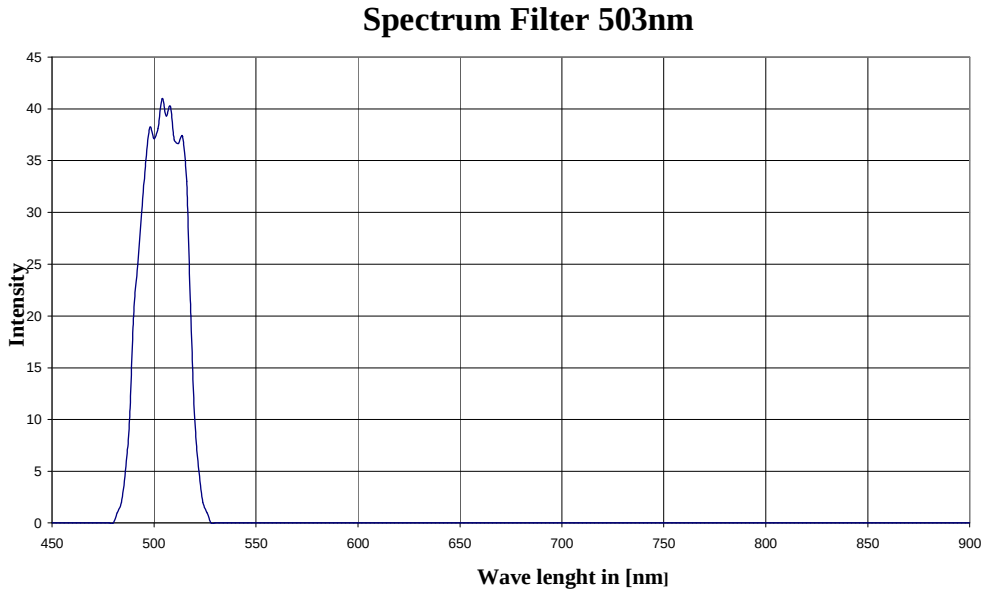

Figure 6: Transmission of the optical band pass filter

Since the band pass filter results in a reduction of the light intensity, it was necessary to think of an active lighting of the scenery. Therefore high power LEDs with the necessary wavelength of 503 nm were placed at the front of the robot illuminating the maize row. With these LEDs it is possible to raise the contrast. Besides the usage of an optical filter it was decided to use the CMUcam3 for this task instead of the CMUcam2 because of its higher performance and flexibility.

Freestyle

The freestyle task is a combined performance of the Amaizeing'09 and the robot "smartNAD" which is also participating in the Field Robot Event 2009. During this task the two teams present the autonomous placement of maize plants (illustrated by artificial maize plants) in the field by the robot Amaizeing'09 and the autonomous watering of these plants by the robot smartNAD. Therefore, an actuator equipped with electrical magnets (see figure 7) was developed to be able to carry the artificial maize plants that were equipped with small metal plates. This assembly can be placed in the back of the robot instead of the weed control actuator. To be able to place the plants at a predefined position the robot can control the electrical magnets to drop the plant while it is driving.

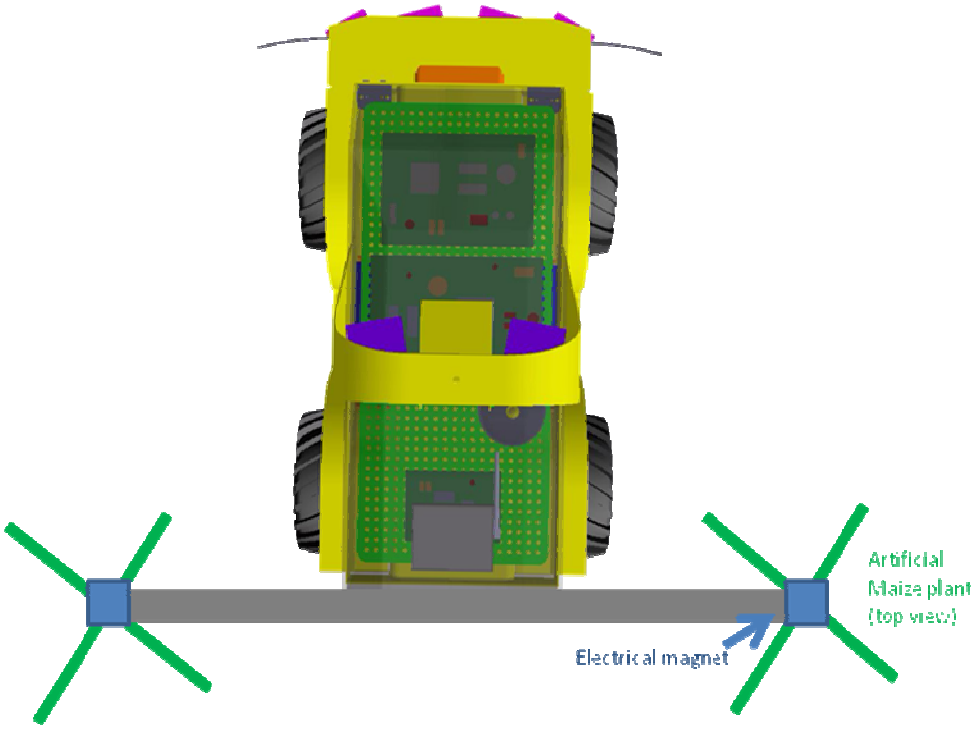

Figure 7: Field robot Amaizeing'09 with freestyle actuator for placing plants

Following this action, the robot smartNAD will detect the dropped plants with its laser scanner and will approach them for the watering of the plants. Therefore, it will use its weed control actuator.

Conclusions

The field robot Amaizeing has been enhanced to fulfil the new requirements of the tasks of the Field Robot Event 2009. Therefore, new technologies including hardware and software were integrated into the existing robot system. The resulting system architecture of the field robot Amaizeing'09 can be found in figure 8.

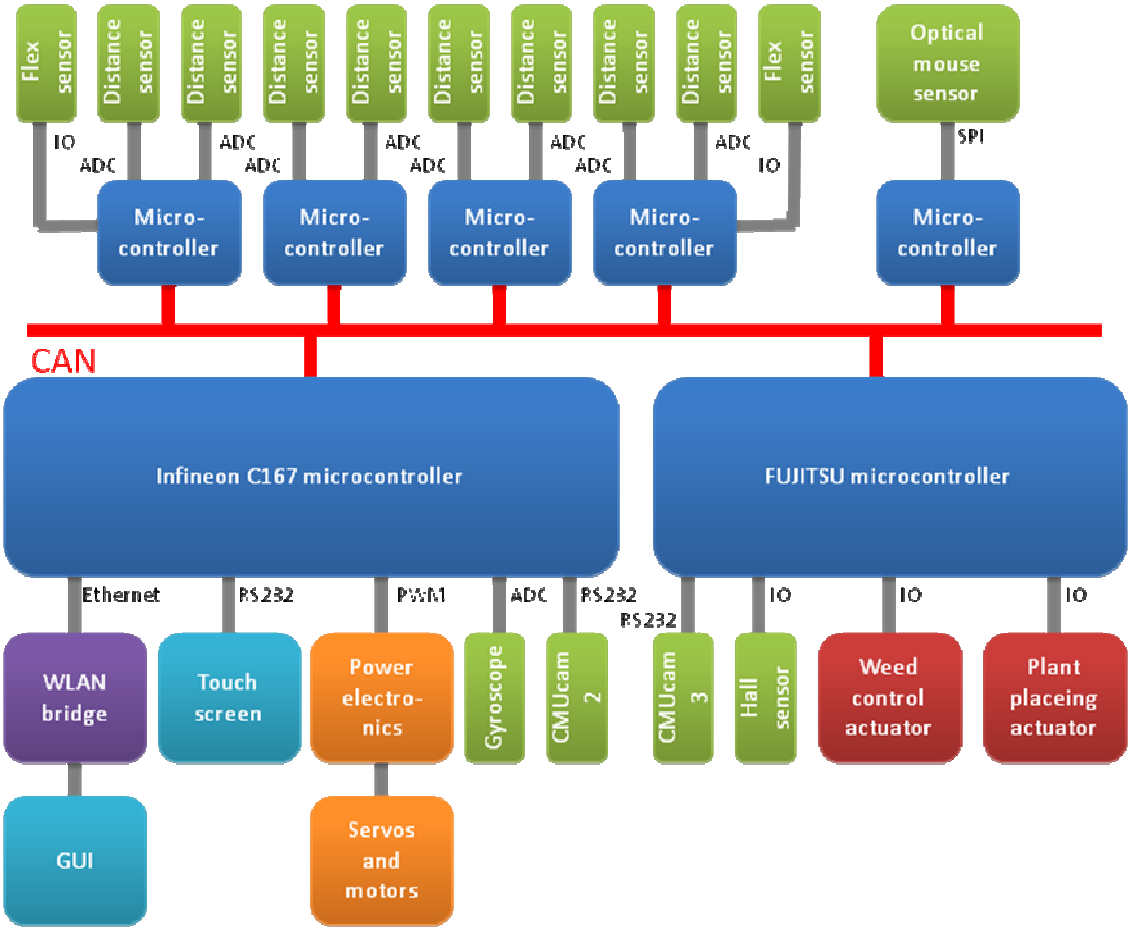

Figure 8: System architecture of the field robot Amaizeing'09

Acknowledgments

The whole team would like to thank it's generous sponsors of 2007 and 2009 and the team of the University of Applied Sciences Osnabrück, especially Ralph Klose, Andreas Linz, Marius Thiel, Arno Ruckelshausen, Jürgen Wübbelmann, and the student dean Peter Roer for shifting an examination outside the FRE.

AMAZONEN-Werke H.Dreyer, Hasbergen-Gaste, Germany

INOEX GmbH, Bad Oeynhausen, Germany

Phytec Technologie Holding AG, Mainz, Germany

Glyn Jones GmbH, Idstein, Germany

Powermagnetshop, MTG Europe e.K., Weilbach, Germany

Graupner GmbH , Kirchheim/Teck, Germany

References

[Agronaut 2008] Brenningmeyer,T., Bruns,A., Conforti,C., Deters,R., Dünninghaus,M., Elberich,M., Flothkötter,T., Greb,R., Heitjan,J., Hohaus,M., Lake,S., Pietruschka,L., Ripke,M., Thoben,J., Voss,H.: AGRONAUT – Autonomous Field Robot, Proceedings of the 6th Field Robot Event 2008, pp. 42-53, University of Applied Sciences Osnabrück/Germany, ISBN : 978-3-00-027341-4, 2009.

[Amaizeing 2007] "Modular sensor platform for autonomous agricultural applications" Steffen Meinke, Andreas Ganseforth, Dennis Hagen, Thomas Eichler, Markus Kreienbaum, Matthias Gebben, Tobias Niermann, Jens Egbers, Stefan Haller, Arne Siebe, Dimitri Willms; Field Robot Event 2007, Wageningen / The Netherlands, Proceedings.

[Eye Maize 2004] "Field Robot EYE-MAIZE" ; Frank DIEKMANN, Jens FLEISCHHACKER, Johannes HENKEL, Ralph KLOSE, Torsten KÖNIG, Martin MEIER, Nicola MOCCI, Axel MÜHRING, Daniel NEGD, Tobias NOLTE, Evert NORD, Maik SCHOTMANN, Johann SCHULZ (Student project supervised by N.Emeis, A.Linz, A.Ruckelshausen); Field Robot Event 2004, Wageningen / The Netherlands, Proceedings, ISBN 90-6754-818-9, March 2005

[Maizerati 2006] "Sensor fusion based navigation of the automous field robot Maizerati" R.Klose, J.Klever, A.Linz, W.Niehaus, A.Ruckelshausen, M:Thiel, M.Urra Saco, K.-U. Wegner, Bornimer Agrartechnische Berichte, Heft 60, 2007, S.56-62.

[optoMAIZER 2005] "Field Robot optoMAIZER" ; Ralph KLOSE, Martin MEIER (diploma thesis supervised by A. Ruckelshausen, A. Linz); Field Robot Event 2005, Wageningen / The Netherlands, Proceedings, ISBN 90-6754-969-X, November 2005

Team BooZer

Sjoerd Brinkman, Jelle Buit, Erwin Jonk, Simon van der Meer, Marco Nehmelman, Pepijn Thissen
Address: Team BooZer, Boeijerstraat 56, 1483 TK De Rijp, The Netherlands

Abstract

This is the final report for the Field Robot Event 2009 from team BooZeR. This report contains all the specifications and our experiences of building our robot. The main goal of the Field Robot Event is to develop and build a robot that is capable of performing multiple tasks within a cornfield.

Our team successfully completed building the robot BooZer, using infrared and GPS sensors. We found the hardest part was the sensor fusion of all the infrared sensors and the GPS receiver. But in the end through extensive testing it was possible, though we had to recalibrate the infrared sensors almost every day because of the change in the surroundings.

Introduction

The Field Robot Event is organized by 'Wageningen Universiteit' in Holland. The idea of the event is to promote the use of robots in agriculture. This new breed of robots must be able to perform multiple tasks, without destroying its surroundings. The tasks, like watering, weeding and planting seeds are considered as human tasks because of the diligence and skill of farmers. The Field Robot Event is an international contest between students from different universities and schools. A special field of maize is grown for this contest providing a steady base for the contest.

For the robot, the organization has no rules, using everything you see fit. The contest itself has some rules though. You must navigate through rows of maize without destroying the plants. The design of the robot is limitless because teams can learn from each other using different techniques.

The main goal is to develop en build a robot that is capable of performing multiple tasks within a cornfield. The tasks represent different qualities of the robot, speed, agility, intelligence and alertness. For the robot, the organization has no rules, using everything you see fit.

Hardware

The hardware is divided into separate components considering the speed of certain calculations and easy replacement. The separate parts all have one function, controlling motors, servos and gathering information from the infrared sensors and GPS. Each part has its own microcontroller running independently so the main program isn't complicated. The parts separated are shown below:

- Intelligence
- Motor control
- Servo control
- Infrared
- GPS
- Spray control

Beside these parts the robot also needs a mobile energy source. Further details can be found in the

chapter 'Power'.

The parts communicate with each other using the I²C bus. This is a synchronized serial communication bus. The maximum speed of the I²C bus is beyond 400kHz, but 100kHz is good for our robot. The Intelligence is the master, requesting and sending data. Slaves can only send data after a request by the master.

The complete layout of the communication between the different parts are shown below:

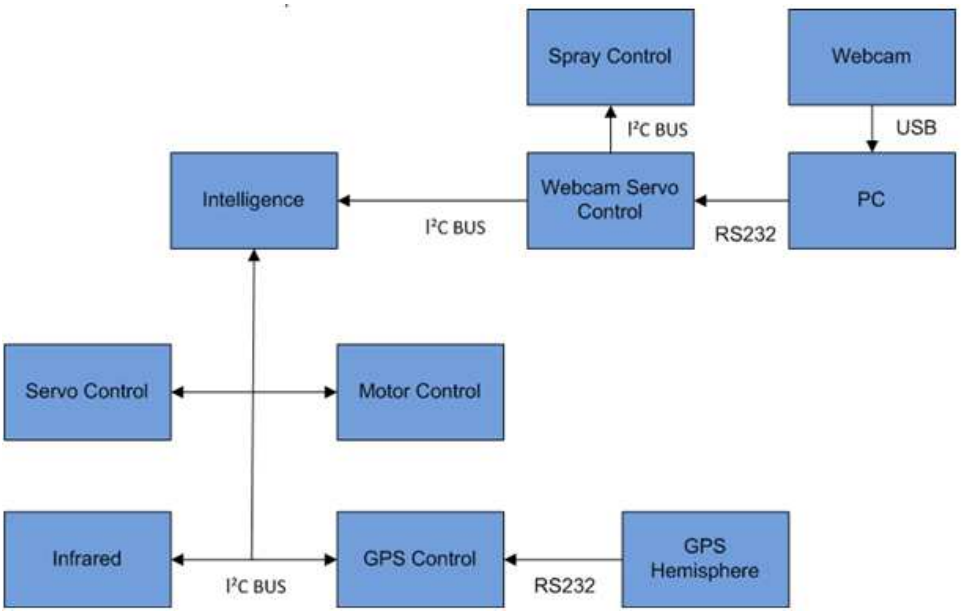

Figure 1: Complete layout of the communication between different parts of the robot.

Intelligence

Hardware: The heart and brains of our robot is an ATmega128 microcontroller. The intelligence is the I²C bus master, providing the whole bus with some pull-up resistors. A LCD and a duo colour led are used for easy debugging.

Function: The function of the Intelligence is to collect and control. Using the inputs, Infrared and GPS the Intelligence steers and moves the robot with the outputs, Motor and Servo control.

Motor control

Hardware: The hardware of the motor control consist of an ATmega32 and 4 LMD18201 fully integrated H-bridges. These are very easy to control, heaving 3 separate lines for speed (PWM input), brake and direction.

Function: The function of the motor control is to ensure that the 4 motors are turning at the right speed and direction. The intelligence sends a command to the motor control every time the speed or direction is altered. The speed can be altered in different ways for each motor separately, all motors or left and right. This last feature is used in corners to lower the turn radius and traction control. For safety reasons the controller can be adjusted to limit the motor speed.

Servo control

Hardware: The Servo control also uses a ATmega32 microcontroller. The controller provides 6V for the servos providing more power. On the servo controller is room for 4 servos although we only use 2.

Function: The function of the servo control is to adjust the servos when needed. This can be done in 2 ways. Adjusting the servos separately or counter steering. Because of our steering construction the robot can make tight corners only when the servos steer in opposite directions. There is also a safety feature for the servo control limiting the turning range.

Infrared

Hardware: The Infrared unit uses a ATmega32 and the 8 build-in 10-bit analog to digital converters (ADCs) to receive a voltage coming from the 6 Sharp GP2D12 IR distance sensors. The Infrared unit is prepared for a lot of interference on this voltage and filters can be mounted if needed.

Function: The Infrared unit has one primary function consisting of sampling the voltage coming from the sensors and taking a average of this value over a period of time and alter this to a more logic value. The actual value of the independent sensor are send over the I²C bus when the intelligence requests them.

GPS

Hardware: The hardware of the GPS consist of 2 parts, one being the Cresent Hemisphere VS110 GPS receiver and the other a interpreter. This interpreter is a ATmega32 with a RS232 to TTL converter. The Hemisphere uses 2 separate GPS antennas to calculate the heading, speed and other variables. Basically it's a very precise compass using the difference between the two antennas instead of the earths magnetic field.

Function: The interpreter translates the serial output of the Hemisphere to I²C very fast. Just like the Infrared unit the heading and other values are send over the I²C when the intelligence requests an update.

Spray control

Hardware: The spray control is somewhat complicated and a system on its own. This is done to ensure the control of the robot stays unaffected while searching for a golf ball. A webcam is mounted in a stand with 2 servos tilting and panning the webcam. This is done by a ATmega32 microcontroller.

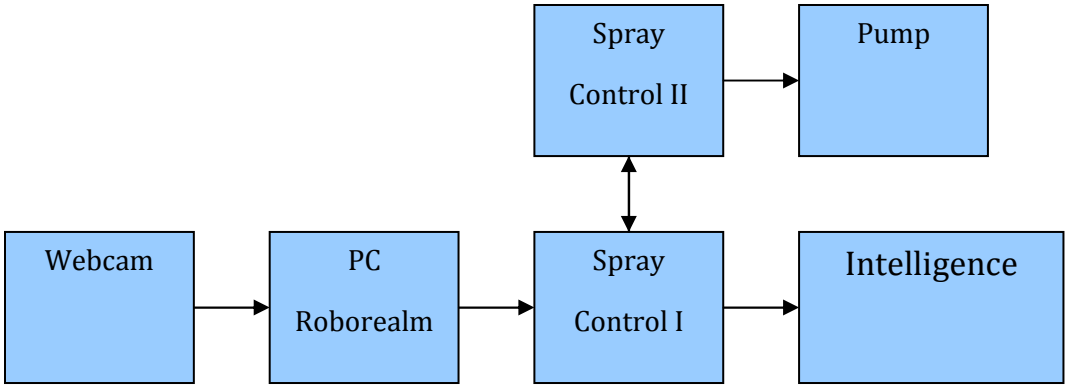

Figure 2: The full layout of the Spray control.

Another ATmega32 controls a pump of which the nozzle is located next to the webcam. The ATmega's, including the Intelligence, are connected using interrupt lines. The pump used is an old windshield spray pump from a Volkswagen Polo.

Function: The webcam is mounted up front on the robot and connected to a laptop/miniPC with Roborealm software. If a ball is found in the image (status: found) the Roborealm software

computes the difference between the location of the ball and the centre of the screen. This is used to steer the servos until the ball is in the centre of the image (status: lock). This is done while the robot is still navigating through the rows. The Spray control stops the robot and activates the pump. After that the robot can move further to detect and spray other balls.

Power

For powering all the components, sensors and actuators the robot carries a mobile energy source. This consists of two 7.2Ah 12 sealed lead acid batteries. Different parts require different voltages, four switching power supply units, based on the LM5005 are used to convert the 24 V (batteries in series) to 5 V, for the logic and microcontrollers and 6 volts for servos. The motors are powered with 24 volts, using the H-bridge controllers. The spray pump also works on 24 V.

Sensors

Multiple sensors are needed to fulfil the different tasks, when it comes to navigation and object recognition. All of the sensors used are listed below:

- Infrared distance sensor Sharp GP2D12;
- Cresent Hemisphere VS110 GPS Unit;
- Logitech Quickcam E2500

Infrared sensors

Sharp GP2D12

- Dimensions: 45 x 15 x 19 mm;
- Detection range: 10 cm to 80 cm;
- Highly tolerant of ambient light and IR interference;
- Nearly immune to variations in object colour and reflectivity;
- Analog output: 2.5V (10cm) - 0V (80cm)

Figure 3: The Sharp GP2D12.

The Sharp GP2D12 consists of an IR emitter and detector combined in one package. Commonly used in robotics, industry and other environments. Its high range, precision, and immunity to interference make it a perfect distance sensor for the Field Robot Event.

Some research was done how to use the output of the sensor and there were two possible solutions found to use the output:

- Compute a formula to transform the output in volts to distance in cm;
- Use the digital output of the conversion (ADC);

After some thinking the first idea was rejected because of the useless transition to centimetres. The output can be used directly using the same sensors and comparing the data from two opposing sensors.

GPS

Crescent Hemisphere VS110

Key VS110 Series Advantages

- Affordable solution delivers 2D GPS heading accuracy better than 0.1 degree rms
- Differential positioning accuracy of less than 60 cm, 95% of the time
- Integrated gyro and tilt sensor deliver fast start-up times and provide heading updates during temporary loss of GPS
- Fast heading and positioning output rates up to 20 Hz
- The status lights and menu system make the VS100 Series easy to monitor and configure

Figure 4: GPS receiver with antennas.

The Crescent Hemisphere VS110 uses 2 antennas to compute the heading. This is computed from the exact positions of the 2 separate antennas. The heading is a virtual line through both antennas and has a value between 0 and 360º.

At first some very difficult and time consuming calculations were done to use the heading to steer the robot. A function was written that could calculate the angle necessary for steering and could update the servos. This led to faults and lots of errors. After some testing a very simple but effective strategy was used in corners and on straight parts.

If the robot decides to take a head turn or drive along the end of rows the actual heading is stored beforehand. The necessary change in heading (180º for head turn and 90º for end of rows) is subtracted from the actual heading. The robot steers in the appropriate direction, and waits till the actual heading meets the desired heading.

The robot itself has no speed feedback of any source, but after a lot of failed corners, this seemed really necessary. Luckily the Hemisphere has lot of features, including actual velocity output. With this we can change the speed in the corners depending on the actual charge of the batteries. The Hemisphere has some interesting features:

- Normal velocity
- Rotating velocity

For determining the speed before a corner the normal velocity can be used. The speed output cannot be used directly but with a formula the velocity can be used to change some variables for making good head turns.

The rotating velocity is left unused, because a combination of the heading and normal velocity worked out great for cornering.

Vision

The third important sensor is the vision part. It consists of a Logitech Quickcam E2500 and vision software Roborealm.

Figure 5: The webcam

Logitech Quickcam

- VGA resolution 640 x 480 pixels
- 3x digital zoom
- 30 fps
- USB 2.0

The webcam is connected to a PC with Roborealm software. This is a vision software that has lots of features when it comes to detecting shapes, colours and light intensity. It incorporates a lot of filters that can be used to clear the image, sort colours and editing other visual aspects. A lot of calculations can be done within the software, to minimize the effort of surrounding soft- or hardware. Roborealm has some interesting outputs since its origin lies within robotics. Its capable of handling servos and multiple serial outputs.

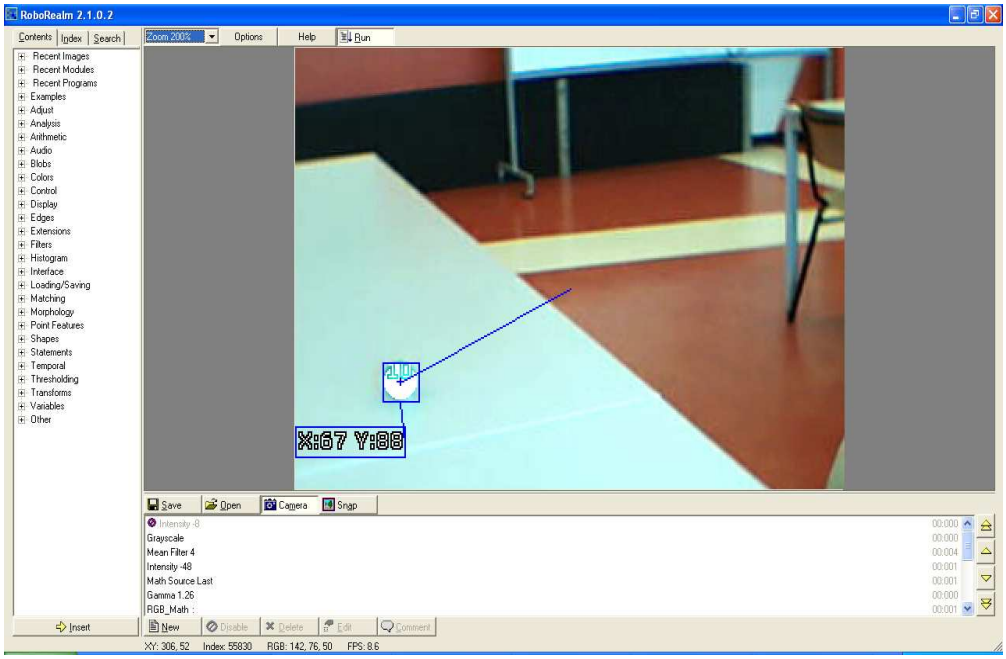

Figure 6: A screenshot of Roborealm with a lock on a golf ball.

Mechanics

Some key features of the robot's mechanics are described separately in this chapter:

- Suspension;
- Steering;
- Frame;
- Sensor mounts;
- Wheels;
- Spray setup;
- Initial changes;

Suspension

For optimal performance the robot should stay stable in rough terrain, ensuring stable sensor outputs. When it comes to designing suspension one has to think about some surrounding factors:

- Weight;
- Terrain;
- Quality;

The terrain and the quality are the most important, the consideration between rough terrain and a super smooth ride are very difficult and can change the design constantly. Working with a high weight, carrying batteries and lots of electronics gives some advantages.

The terrain is made up of loose soil or sand. Combine this with rain and there is a very muddy surface. The variation in height is very minimum in dry conditions and only consists of tracks (of other robots) and some lumps of soil. The surface becomes flat in wet conditions.

Some options are available:

- Independent wheel suspension;
- Independent axle suspension;
- Front or rear axle suspension;

The list ranges from very difficult to fairly simple.

Independent wheel suspension gives the smoothest ride but is very difficult and is mechanically the most heavy construction. Using this method of suspension would make four wheel direct drive impossible. The suspension has to be fairly sturdy to withstand the forces from the heavy frame, this makes it even more difficult.

Independent axle suspension is a lot simpler and provides a nice ride. The main advantage is the fixed axle where the motors and steering can be constructed without the difficulties of a lot of moving parts. The only disadvantage is that the frame will move quite a lot if both wheels on one axle travel over bumps at the same time.

Front or rear axle suspension is only one half of independent axle suspension and gives the same advantages, only the frame (and the axle fixed to it) will be victim to bumps and pits. The main advantage is that the fixed axle can be used for mounting motors without moving parts.

From our point of view four wheel drive and front and rear steering are very important for optimal results. Combining these two things could be done by fixing the motors to an axle and rotating this for steering. A lot of moving parts would be necessary if independent wheel suspension was applied. Fixing the axles would result in a lot of stress on the frame and could result in catastrophic failure. So we applied independent axle suspension:

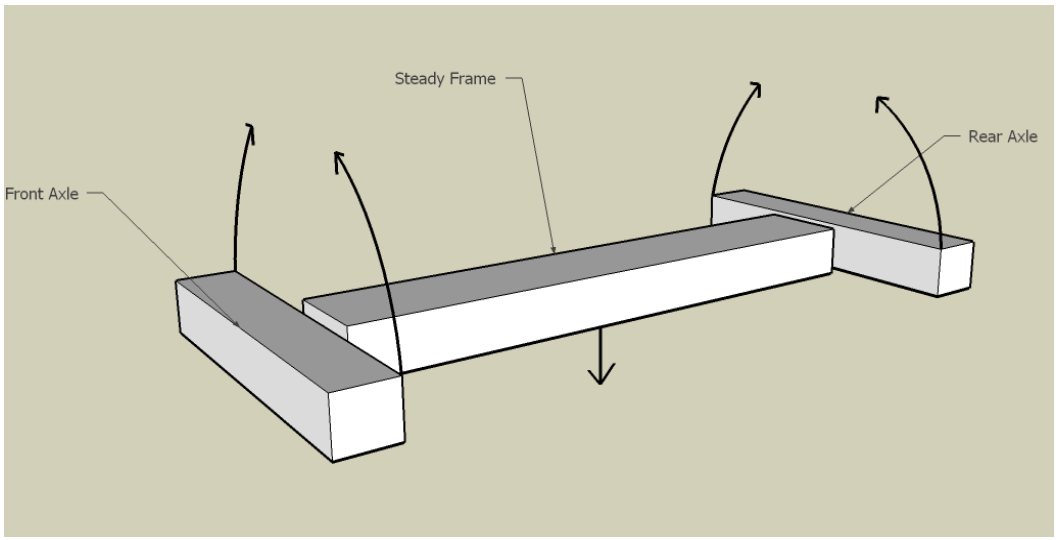

Figure 7: A very simplified version of our suspension. Both of the axles can move independent of the frame.

The main weight is applied on the frame (batteries, electronics, water reservoir, sensors) so the frame stays steady. The axles are fixed to the frame using coiled shocks used in RC cars. These shocks limit the movement of the wheel to about 25mm. This is enough for most lumps and bumps.

Steering

The robot must steer sharply, swift and precise. The robot should be able to make head turns from row to row within a 1.5m area. Our goal was to do this without backing up because many fault can

be made doing that. Using this in combination with four wheel drive seemed difficult at first but fixing both wheels at one axle could solve that problem.

There are some options for steering:

- Steering each wheel separately, using four servos;
- Steering 2 wheels together, one or two axles;
- Steering 4 wheels using 1 servo;

When it comes to steering each wheel separately using four servos the cost of the servos come in to play. First of all four servos are expensive and the servos take all of the beatings of rough terrain hitting objects and tension from the weight. Secondly, controlling four servos at a time could result in serious problems. If one or two servos steer somewhat different then the whole construction could tear itself apart.

Fixing two wheels together gives some problems to the angle of the wheels are the same causing the outer wheel to skid and losing traction. This can be solved using different speeds, rotating the outer wheel faster than the inner wheel.

Using just one servo to steer all wheels is a very complex construction, but is very efficient and steers real nice. This is only possible using fixed axles otherwise the linkage system would break and buckle. This design is mainly used in RC cars.

With the key features in mind fixing two wheels together seemed logic. Although using four servos was a nice alternative, with our budget and microcontroller capacity two servos was a good choice. Having two identical axles made the fabricating more easy, only mirroring the construction. To incorporate four wheel drive a motor mount had to be developed. This mount is used to connect the motor to the axle and the steer linkage.

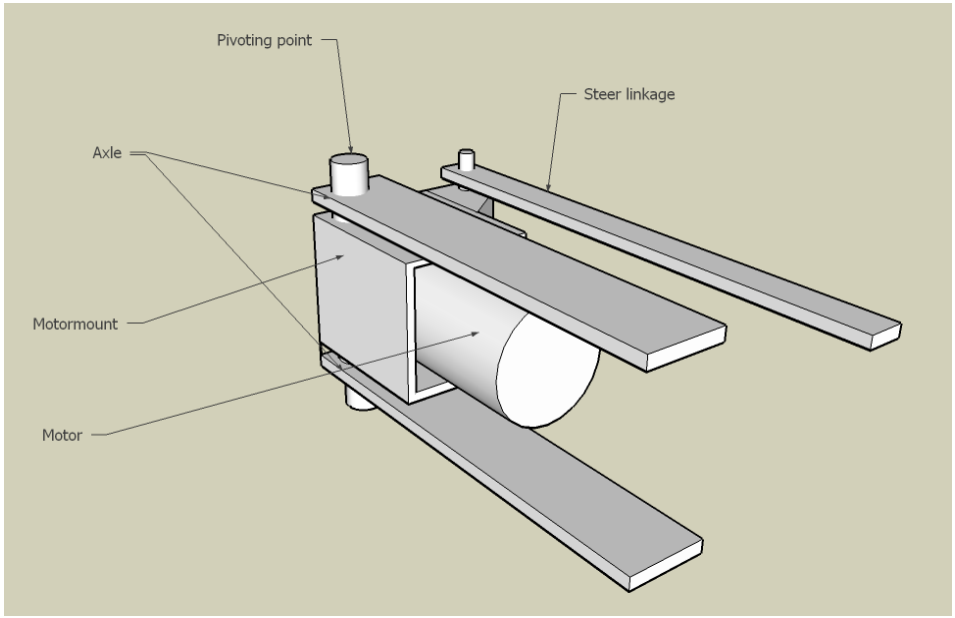

Figure 8: A very simplified version of our motor mount. The motor mount is supported by two pieces of metal, from above and below. The axle of the motor is in the center of the motor, so the pivoting point had to be moved slightly to the side.

The servos used for steering are HITEC HS-805BB+ servos capable of handling 25Kg. The servos

are mounted to the axle and connected to the steering linkage. The link between the servos and this steering linkage is made very weak on purpose. If some external forces were to move the wheels the link would break first, protecting the servo. The servo is mounted upside down because of its size. A mounting bracket made from a piece of aluminum tubing secures the servo to the axle.

Frame

The frame of the robot has to be light weight and sturdy but also large enough to bolt other components on to it. This frame should hold the axle for the pivoting point of the front and rear axle.

Not much consideration went into the frame. A aluminum beam and some solid pieces were used to create the frame. The solid pieces are slotted into the ends of the beam, creating a closed frame. The axle for pivoting the axles runs through the whole frame being held by the solid pieces.

The beam is quite thick, and very suitable for mounting other components like the frame for the shocks. This frame is constructed out of aluminum structural profiles. These profiles can be mounted together using nuts which slot into the profile. In this way the height of the shocks can be easily adapted, for softer or sturdier suspension.

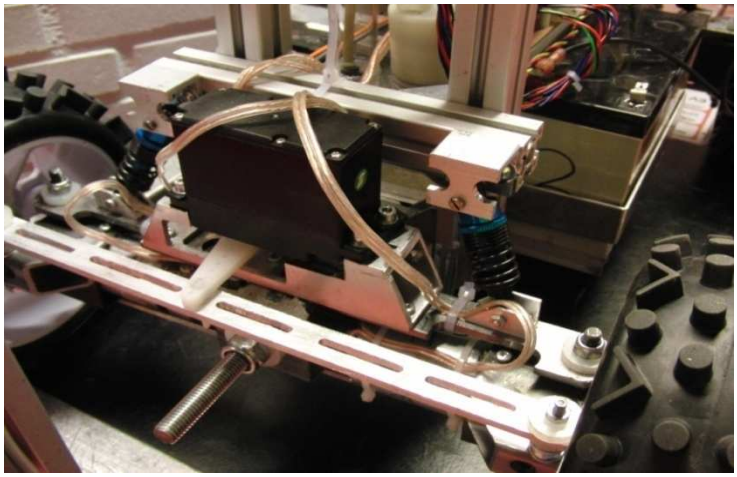

Figure 9: A photo of the final design. Most of the vital parts for steering, suspension and axle are visible. The servo, shocks, structural profiles and the motor mounts.

Sensor mounts

The sensors should be connected to the frame, but must able to move around. The height of the IR sensors may be adapted, to ensure good sight on the maize. The GPS antennas must be fixed to the frame, on top of the robot standing more than 50cm apart. This to ensure the precision of the heading.

The frame used for the shocks is used to mount the sensors, standing tall on the robot, providing room for another aluminum beam. The GPS antennas bolt onto this beam and the IR sensors, mounted to another piece of structural profile slot into this beam. The Hemisphere itself is located underneath the beam.

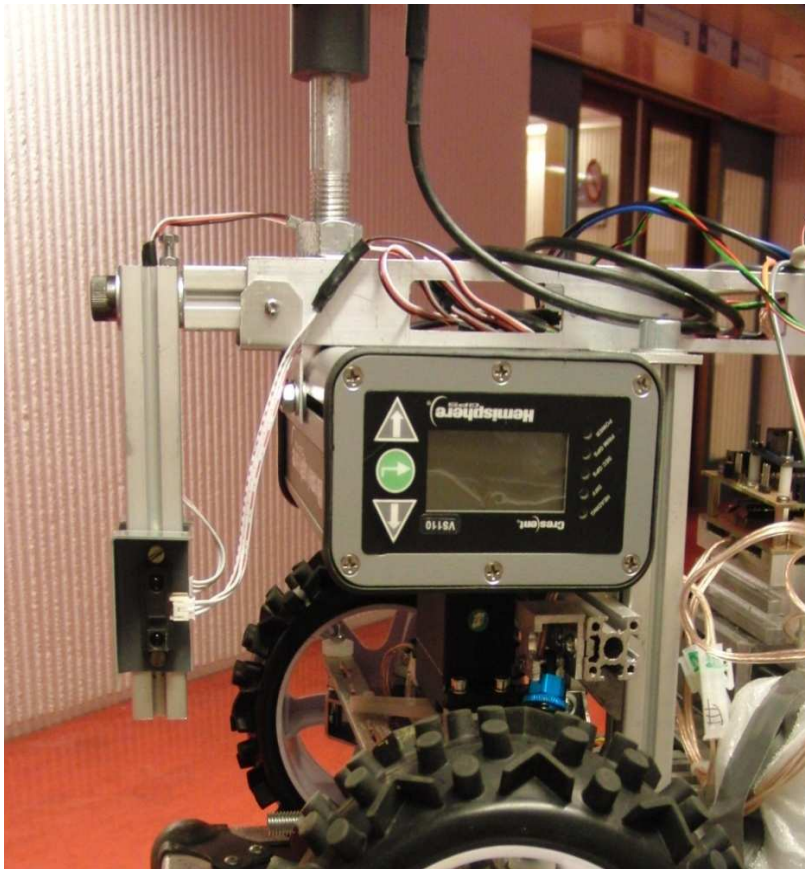

Figure 10: A photo of the rear sensors, mounted onto a piece of structural profile housed within a plastic shield. The structural profile is fixed to the beam by the bolt of the GPS antenna (not fully visible). The IR sensors can be fitted anywhere on the piece of structural profile, adjusting the height

Wheels

Moving around on muddy or loose soil is difficult enough, but even harder when trying to move around a heavy robot. The choice of the wheels wasn't easy and getting the right ones was even more difficult.

The perfect wheels for a agricultural robot that is able to drive on different surfaces depends on some factors:

- Type of surface
- Weight of the robot
- Amount of grip needed

These are the main factors, but there are more things that must be considered before buying wheels. The width and height of the wheels are very important when it comes to handling and steering the robot.

The final wheels were found after hours of searching on Ebay with 4 people in different sections of Ebay. These wheels completely match the 'ideal' wheels factors.

Spray setup

One of the tasks is to detect and spray water on green golf balls. This has to be done with great precision to earn more points. These green balls are detected using a PC and a webcam. To allow

easy precision while hitting the balls a very neat solution was found. The whole spray setup is divided into some parts:

- Webcam setup;
- Spray pump/water reservoir;
- Spray control;

For more information about the spray control, see chapter 2.6.

If a ball is in range, Roborealm will guide the webcam till the ball is in the center of the image. This is done by 2 servos, tilting and panning the webcam. A nozzle placed next to the webcam is mounted at a slight angle, ensuring the jet of water coming out of the nozzle hits the ball in the center.

The webcam setup is made out of aluminum tubing and some other aluminum profiles milled down for weight reduction. A frame is constructed around the webcam clamping it firmly along with the nozzle and screwed into the frame. This frame is supported by a servo at one end and a ball bearing at the other. The panning action is performed by another servo mounted beneath the other servo.

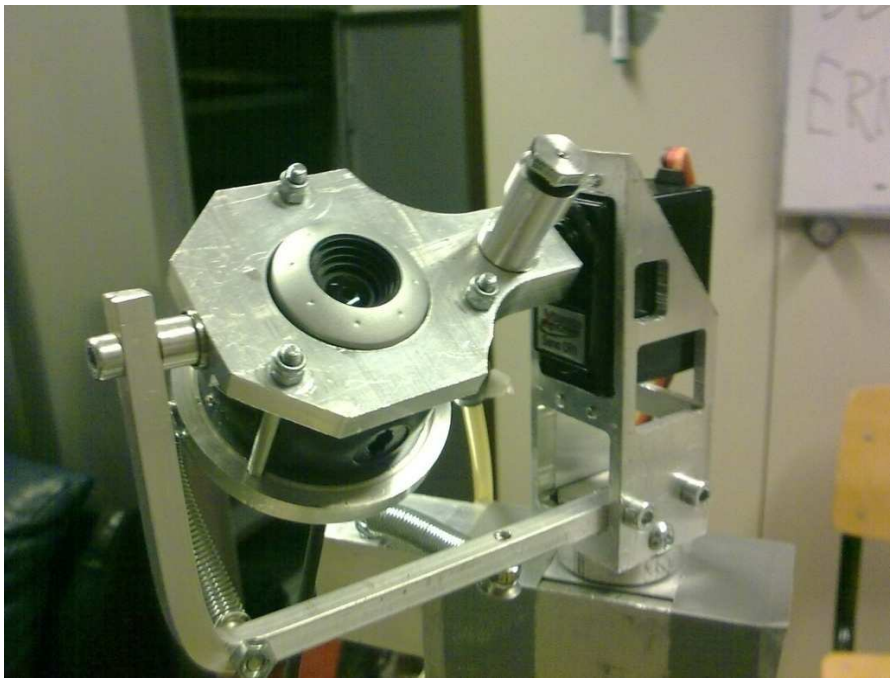

Figure 11: The webcam setup. The panning servo is mounted below the aluminum frame, inside the beam. Springs are attached to counteract the play in the servos.

To create a jet of water an old windshield spray pump is used. This pump is housed within a reservoir holding up to 0.8l. This reservoir is custom made to fit the robot, on top of the frame between the batteries.

Software

Most of the software is written in C using Atmel AVR Studio 4.16. Each part of the robot has its own microcontroller and therefore its own software. We chose this approach because then we could take as much calculations as possible out of the main microcontroller. So the main intelligence microcontroller collects the data from the other microcontrollers, processes it and then sends the correct speed and heading to the motor en servo controllers.

I²C

As base communication method we use I²C. I²C uses a principle of one (and sometimes more) master controller(s) and one or more slave controllers. Our robot uses one master called the intelligence (More information about the intelligence is in the next paragraph). And we are using multiple slave controllers. The idea about the I²C is that every controller in the I²C network gets its own address. Using this address the master controller can read and write data to a slave controller.

Using this principle we operate our robot. The master collects data from the slave controllers which are continuously storing data from there sensors. Using this collected data the master controller commands the controllers which are operating the robot.

For the master there are two base functions. One for reading data from a specific controller and one for storing data on a specific controller. The software for the slave side is interrupt driven. This interrupt is fired when a connection is made to the address of the slave. The interrupt handles the different states of the I²C communication and data can be transmitted or received.

Intelligence

This software consists of a few C files. One for every task, and one with all the common functions used in all tasks. Each task is divided in cases. For every situation the robot can find itself in.

All tasks use a fuzzy logic heading calculation function which takes every infrared sensor input and gives weights to each input. The GPS heading is also taken into account. The ones that seem more important at the moment get heavier weights then the ones that seem less important. Every weight is then put in a formula which gives the heading it thinks will be best. This calculation is done approximately 300 times per second so the robot can react very fast to changes in its surroundings.

A few other functions are for example the one that calculates the best moment to turn into the next row and when to stop turning. There is also a function that reads the GPS heading, which gives the current heading in degrees.

Motor control

The motor controller uses one of the internal timers of the microcontroller to generate a 1 kHz PWM signal. The master can send the speed, and a number which corresponds to one of four motors. This will change the speed of the motors instantaneously.

Servo control

The servo controller is much like the motor controller. It too uses a timer to generate a PWM signal but this time a 50 Hz signal to drive the servos. The master can control them in almost the same way as the motors. The master sends a heading, and a number corresponding to a servo.

Infrared

The six infrared sensors are connected to six of the eight analog digital converter pins of the microcontroller. The ADC values are read and processed to make them easier to use for the master.

GPS

The main goal for this software is to store GPS data and make it available for the intelligence to read this GPS data. This software makes two ways of communication possible. The first is the serial communication between the microcontroller and the Cresent Hemisphere VS110 and the second is the I²C communication between the GPS microcontroller and the intelligence microcontroller.

The serial and I²C communication are both interrupt driven. For the serial communication this means that when the microcontroller receives data on the RX pin an interrupt is fired. Using this interrupt the data is stored and analyzed for its usability. After this the data is stored again so it is made available

for read out via I²C communication.

Spray control

The microcontroller is connected to the webcam computer via serial connection. The microcontroller reads the coordinates of the golf ball via this connection and calculates the correction for the servos, so the spray nozzle is pointed at the golf ball. The servos are controlled by a 50 Hz PWM signal which is generated with a timer.

Conclusion

We have build a robot on a very low budget and this helped us to be very creative in the design of the robot. We had to build our robot from cheap parts or parts we could get for free. We managed to build a solid robot still with a lot of strengths and little weaknesses.

The strengths of our robot are the 4-wheel drive and 4-wheel steering, they give the robot a combination of high traction with tight steering. We also keep the navigation simple which makes it robust and allows the robot to drive relatively fast. We also have a strong power supply consisting of two 12 volts lead acid battery combined with a switching regulator we can go quite far on one set of battery's. Due to overpowering the motors the robot can reach pretty high speeds.

Our weaknesses are that we might be a little unprepared for bad weather. Due to our low budget we didn't manage to get a better camera for ball detection than a standard webcam which is build to work inside and not outside.

Overall we made better progress than expected and we didn't lose too much time in doing unnecessary complicated stuff. And we managed to get equipment from several companies which really helped us out.

Acknowledgments

We would like to thank our sponsors. Without them the whole robot wouldn't exist in its current form:

All the professors who have helped us:

Gerard Baas, Toon van Griethuijsen, Cees Keyer, Jan Meijer, Erik Steuten, Martin Stolk

Everyone in the workshop of our school.

And Dalkia for providing us with batteries and fresh air.

References

All the information we needed we got directly from our professors, and our sponsors.

The Autonomous Robot – CornStar project

Miran Lakota, Peter Berk, Jurij Rakun, Peter Lepej, Tomaž Paripovič
Faculty of Agriculture and Life Sciences, Biosystems Engineering, Pivola 10, 2311 Hoče, Slovenia
E-mail: {miran.lakota, jurij.rakun}@uni-mb.si

Abstract

In this paper we present a small automated self oriented mobile platform, which could one day replace human labour on the fields, by doing the work quicker, with higher precision and lower costs. To achieve this goal we equipped the platform with ultrasonic sensors, high resolution digital camera and an onboard embedded computer. We tested the robot to detect simple objects, where several simulation runs proved promising successful rates.

Introduction

In the age of technological revolution agriculture will be one of the disciplines that will probably benefit most in the future. As big food producers relay on the use of heavy machinery, this is still not the case for middle and small farms, which can pose a potential food safety problem. If handled manually, food can transmit disease from person to person as well as serve as a growth medium for potentially harmful bacteria. Nevertheless, some work still demands manual labour that is time consuming, exhausting and expensive. The thought of introducing a small army of intelligent robots to do the job quicker and more accurate seems appealing, but we are not just there yet. For one, natural uncontrolled environment poses a challenge with its changing conditions. An overview on the subject showed that there are some potentially good solutions but the authors relay on specific conditions (like night time) or their solution is designed to work in controlled environments (green house) and some are simply too big or too heavy to be useful at this stage. In this paper we try to tackle the problem by introducing our own mobile agricultural platform.

In order to achieve our goal, we decided to put our efforts to build a small autonomous self oriented mobile platform, that could for instance serve as a potential tool for selective pesticide spraying, fertilizer applicator or even as a device that could estimate the yield at the end of the harvest by simply taking digitalized snapshots of the tree canopies. In the following section we start by describing our solution in detail, continue with the results and a short summary that will be our cue for the future.

Description

Our platform consists of four crucial components. The first is an onboard embedded computer with high speed digital camera in a wireless connection. The second is a four-wheel drive that makes possible for platform to move around the rough terrain. The third are the ultrasonic sensors that help to keep track of the surroundings. And finally, the fourth part, the onboard reservoir and nozzles that spray the plants.

Figure 1 depicts the platform in detail along with all the components and their location.

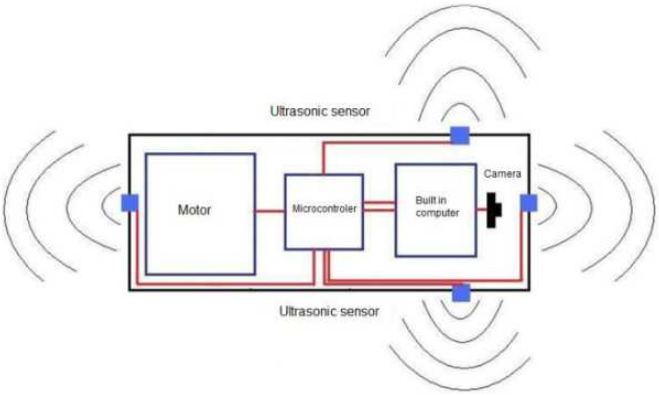

Figure 1: Mobile platform components.

Microcontroller circuit

The job of controlling the mobile platform is left to the microcontroller circuit that receives instructions from the embedded computer and acts according. It also consistently reads distance measurements from all four sensors and transmits the data to the embedded computer via serial (RS 232) connection.

The microcontroller also has one important job. In case of an obstacle on the path of the mobile platform, the platform stops without waiting for the instructions from the off-field workstation. The platform detects the obstacle if it is less than one meter from the platform, so it can stop safely.

The microcontroller part is realized with the help of PIC16F877A microcontroller, which seemed perfect for controlling the servo motors (with its build in PWM circuit) and for gathering measurements from the ultrasonic sensors (by using GPIO lines).

Distance measurements are captured with the help of four SRF04 ultrasonic sensors that can measure a distance of up to 3 meters, which is more than enough. Distance measure from each sensor is captured four times a second.

Embedded computer and data transfer

The basic task of embedded computer is to gather all relevant data from onboard sensors and transmit them to off-field workstation. There the data is processed and a decision is made on how to control the platform (which direction to take, when the open the nozzles, speed adjustment, etc.). Of course most of the decision making algorithms could run on embedded computer, but are still located on the off-field workstation. The purpose behind this scenario is easy and rapid development of new algorithms as well as almost instant control of the platform in cases of emergency.

The onboard embedded circuit can be divided in to four components. The first is a fast, high resolution digital camera that captures video stream and the second a microcontroller that controls the servo motors and captures distance date from the ultrasonic sensors. The third component represents a wireless interface that works according to the IEEE 802.11g standard and the final, fourth component, an embedded computer. For the last we selected a small size, x86 compatible computers, that consumes only around 15 W (load) and still manages to compute around 1400 MIPS or 1200 MFLOPS. The embedded computer is depicted on figure 2.

The onboard embedded computer and the microcontroller are connected via serial (RS232) interface at a speed of 2400 bps, while the wireless connection operates at 54 Mbps to provide off-field workstation with distance data and a video stream, which demands a wideband wireless connection.

As the operating system for the embedded computer we have chosen and customized a version of

the Linux operating system (Debian based). The first step we took was to remove all unnecessary software. As the second step, we compiled a custom 2.6.22 kernel according to the hardware specifications and finally, as the third step, we tuned the settings for the TCP/UDP protocols to reserve more buffer space and to transmit small data chunks faster in effect minimizing the transmition delays.

Figure 2: An embedded computer - VIA EPIA PX10000 (size: 10 x 7.2 cm).

Server algorithm

Current role of the embedded computer is to act as a mediator between the microprocessor and the off-field workstation. It listens to the incoming distance data from the microcontroller, verifies the accuracy and transmits the measurements by using a wireless connection to the off-field workstation. In the opposite direction, from the off-field workstation to the microcontroller, packets containing instructions for steering and controlling the nozzles are transmitted. Again the accuracy is verified before transmitting the packet to its destination. Additionally a simple “echo” mechanism is provided to check if both communication peers are running as depicted on figure 3. Packets in both directions follow the following form:

ID	Value	Checksum

.

Beside already presented role, the embedded computer also works as a video server for the off-field computer. It captures live stream at the rate of 30 images per second with resolution of up to 1024 x 768 pixels.

In order to be able to capture video stream and to transmit last captured image simultaneously two memory locations are reserved. The algorithm first chooses one of the locations and it stores new image. Then continues using the second memory location and stores another newer image. This is repeated until the off-field workstation requests a new image. In this case, the location of last fully saved image is protected so it is not overwritten by the video stream capturing thread unit it is not fully transmitted. When the server thread locks the memory location, video capture thread stores new images only on one location that is available for writing. This way we provide up-to date data and also prevent the transmition of incomplete images. The mechanism for capturing and transmitting images is depicted on figure 4. Time complexity for all procedures in this algorithm is estimated to be linear.

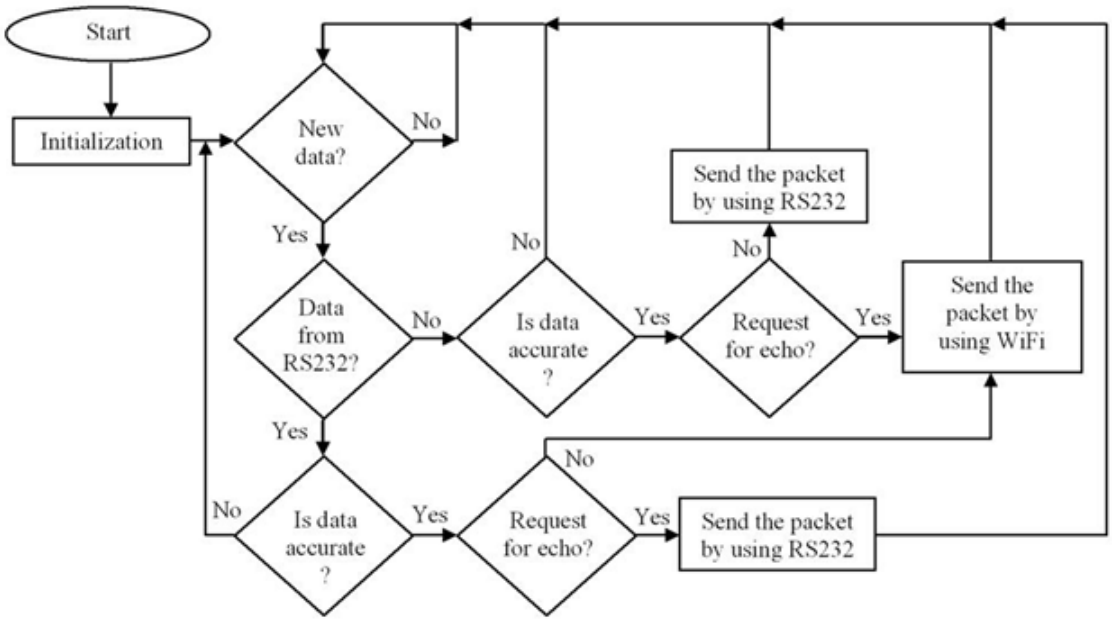

Figure 3: Procedure to relay sensor and control packets.

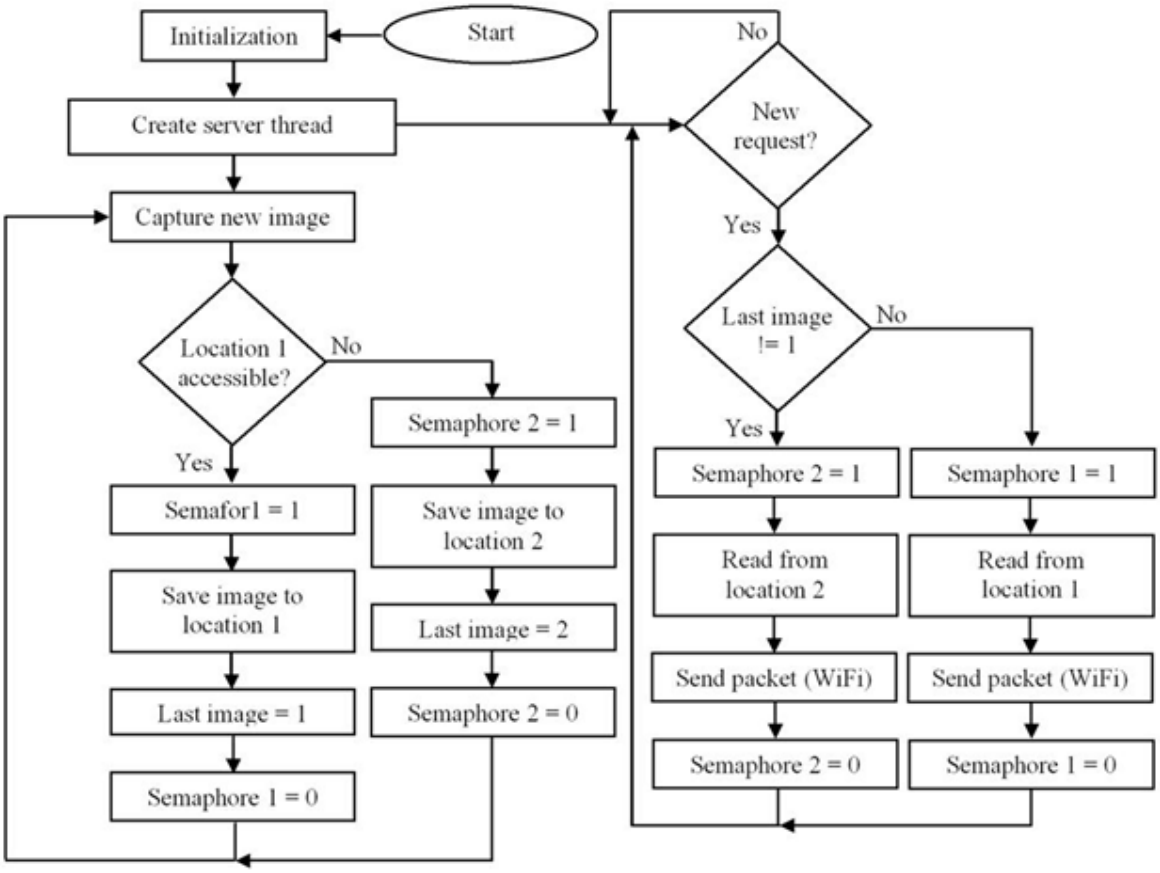

Figure 4: Procedure to capture and transmit video stream.

Distance data

As the distance data is represented by the sensor with one of the values ranging from 0 to 255 values, it is first converted to real distance measurements. By using distance measurements a

decision is made according to eq. 1 and eq. 2 on how to adjust the heading:

$$RA = \frac{SV - RZ}{2}, \qquad 1$$

where *RZ* represents distance between the sensors mounted opposite of each other, *SV* the current value of the distance and the preferred value *RA*, based on which heading correction is carried out according to eq. 2:

$$f(RA) = \begin{cases} Left : SV > RA \\ Right : SV < RA \\ Straight : SV = RA \end{cases}. \qquad 2$$

Figure 5 depicts the flow chart of the distance processing algorithm, which is able to control the mobile platform even in the event that one of the crop rows is not fully planted. *M* on figure 4 represents a value that is measured by the sensor and converted to real measurements. In case the value of M is twice the value of *RA* or more, the crop line at that side is incomplete and the algorithm tries to compensate by comparing the measurements on fully planted side of the robot. In case both values, *LS* as well as *DS*, are greater than *RA*, the robot reached the end of the crop line and stops. The estimated time complexity of the algorithm is linear.

Image processing

The image is received by the off-field workstation using a wireless connection. As we want to detect object on the ground there is no need to process the whole image. So we decided to trim the image only to the lower half of the image. By doing this, we also speed up the transmission and object detection process. Furthermore the image is encoded in Bayer format omitting the need to transmit all three color planes.

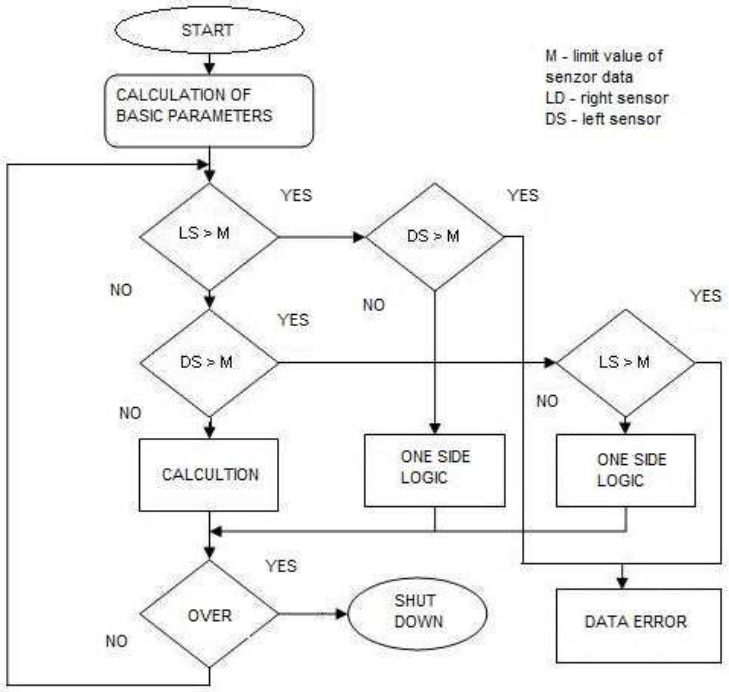

Figure 5: Algorithm for processing distance measurements.

After receiving Bayer encoded images, they are converted to HSI color space. In contrast to RGB

color space it proves to be much more useful as we can simply segment the image by looking at selected color shades on hue plane. This produces a binary result where we look for regions with potentially right shapes by measuring distance from the gravity point to the edge for each region on the binary image. Figure 6 depicts a test case scenario where we were looking for yellow balls between the lines of corn.

The complexity of the algorithm is estimated to be in range of O (n^2).

Figure 6: Test case scenario (left image) where we were looking for green plants and yellow referential objects. The result of image segmentation is shown on the right.

Steering

In order to drive the mobile platform, we have chosen a high-performance brushless motor (X-power eco A4130-06BL). Technical data for selected brushless motor can be observed in Table 1. This is a three-phase motor so in order to use it, we have added a controller (x-power professional 70-3P BEC), which controls the coils of the motor. The specifics of the controller are depicted in Table 2. The controller automatically detects the number and type of batteries used. In our case we used Lithium batteries (LiPolice 5000mAh/2S 7,4V 75/100A) with two cells. The controller also controls the speed and rotation direction of the motor. Brushless motor speed is controlled with the help of pulse width modulation. At given (high enough) modulation the motor will run at constant speed consuming around 40 watts of power.

Table 1: Brushless motor - Technical data

Cells Li-XX:	2 – 6
U/min/V (without gear):	510
Lenght (without shaft):	65 mm
Weight:	400 g

Table 2: Controler – tehnical data

Cell number Li:	2 – 3
Current, continuous/burst:	70/85A
Dimension:	75x28x10 mm
Weight with cable:	54 g

Results

In order to test our mobile platform we carried out a few simulation runs where we tested the ability to steer and control the platform remotely and to detect small referential object – yellow tennis balls.

First, we tested the platform in a controlled environment, with good artificial light conditions and with no interferences (like occluded regions). The simulation was carried out five times using different spatial distribution of color balls. The results were promising, where the robot steered perfectly and we were able to detect and selectively spray on all referential object.

In the second case, we tested our mobile platform in an uncontrolled natural environment. For the task we selected University's orchard where five more simulation runs were performed following the same procedure as before. Again, we were very satisfied with the results as we managed to detect 93 % of the referential objects. By looking at the captured video stream we concluded that partial occlusion of some of the objects was to blame for slightly lower success rate, where the object were occluded by plants more than 50 %.

Transmition delays

One of the crucial points when working with wireless are of course transmition delays that occur during transfer. We decided to measure it and determine if it could pose a problem. We carried out five tests where we transmitted one hundred images with the size of 270 KB. The results are depicted in Table 3.

Table 3: Transmition delays for five simulation runs.

Average [s]	Maximum [s]	Minimum [s]
0.38	1.21	0.10
0.41	1.28	0.11
0.35	0.97	0.07
0.41	1.02	0.09
0.32	0.99	0.08

On average it took about 0.4 seconds to transmit each image, in worst cases around a second and in best only 0.1 seconds. Looking at the data, we noticed that the worst delay occurs at the beginning of each transmition and once it is transmitting, the speed increases. Never the less, the sequence is good enough to reconstruct the whole path with no missing regions as the images overlay with the same content in at least one fifth of an image.

Conclusion

We achieved our goal and built a prototype of the mobile platform that could serve as a small agricultural tool for various tasks. Of course this is only the first step toward building a universal tool that could do the everyday work quicker, more precise and without human interaction. The first simulation runs proved the concept with more than satisfying results.

Our focus in the future will be on how to improve the algorithms to be more precise and robust. At this stage we can detect only simple shaped object with distinct, prominent colors. In case the object is occluded or has uneven lighting conditions we quickly run into problems. Homomorphic filtering followed by texture analysis therefore seems appealing for future research.

As we want to use the presented platform outdoor in sometimes harsh natural environment we must account for changing weather condition, rough terrain, obstacles and last but not least the interaction between the platform and the human worker. In the future, when we will be satisfied with developed algorithms, we will remove the need for off-field workstation and transfer the algorithms directly to the embedded onboard computer, making the robot totality autonomous.

More information about the robot:
Email: *miran.lakota@uni-mb.si*

Website: *www.fk.uni-mb.si*

References

J. F. Thompson, J. V. Stafford, P. C. H. Miller, *Potential for automatic weed detection and selective herbicide application*, Crop Protection, vol. 10, pp. 254-259, 1991.

R. C. Gonzales, R. E. Woods, 2001. *Digital Image Processing*, 3rd edition, Upper Saddle River: Prentice Hall PTR.

L. Shapiro, G. Stockman, *Computer Vision*, Prentice-Hall, Inc. 2001.

Bulanon, D.M., T. Kataoka, Y. Ota, and T. Hiroma. *A Machine Vision System for the Apple Harvesting Robot*, Agricultural Engineering International: the CIGR Journal of Scientific Research and Development. Manuscript PM 01 006. Vol. III.

Guo Feng, Cao Qixin, Nagata Masateru, *Fruit Detachment and Classification Method for Strawberry Harvesting Robot*, International Journal of Advanced Robotic Systems, vol. 5, no. 1, 2008.

Cratos

Proceedings Field Robot Event 2009 Fontys University Venlo,

Project members: Bakker, Roy; Kersten, Tim; Verstraeten Bob; Vught, Maikel; Weerts, Harm

Introduction

The Field Robot Event 2009 would be the second time The Fontys Field Robot Team would participate in the event. It was decided that the robot that was used the previous year would be re-used this year. This was mainly because we believed the principals and constructions that realized in this robot where good. Also, to start from scratch with the limited amount of time available would be somewhat of a gamble. To work with the available robot would only mean (a lot) of fine tuning.
The challenge this year was that the time given was halved as well as the number of team members. Therefore the decision was made to only adjust the most critical problems of last year. The key problems would be: vision, ultrasonic sensors, the motor controllers, EMI and vibrations in the embedded board.

Electronics

The electronics were entirely custom designed from the supply print to the I2C control bus. This had as great advantage that everything was adjusting to the specific needs of the project. This way a great amount of power and components could be saved. The disadvantage was that last year all kind of adjustments where made and additional features added. This made debugging and analysis of the system extremely complex.

The entire electronic layout was therefore removed and redesigned. Instead of individual supply PCB's for the different power supplies and data PCB's there where now only two prints in the entire robot. One print for the power supply voltage and one for the data communication (I2C). The number of cables and connectors was therefore drastically decreased in comparison to last year's layout.

The power supply delivered all the required voltages in the robot. The required power was also reduced to only 5V, 12V and 24V for the brushless DC motors (separate circuit). In Figure 0.1 the electrical schema is displayed. This schema shows the different power supplies as well as the data communication.

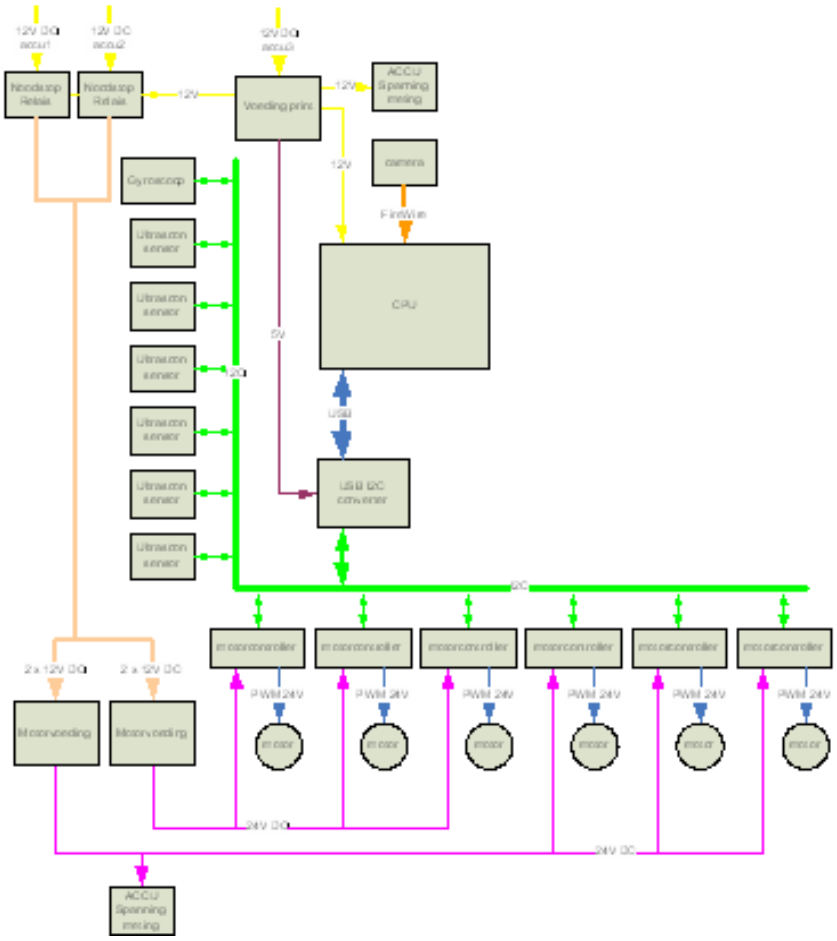

Figure 0.1 Electronical layout

The new print layout and cabling in the robot also solved the EMI problems that occurred during the tests. The EMI was mainly caused by the different cabling, by separating the different supply voltages and by shielding the signal cables these problems where solved.

Motor controllers

Another big problem we had last year where the motor controllers. It appeared that the controllers that where supplied with the motor could not control the mother with low speeds. This proved to be extremely troublesome especially during turning.

The easiest solution to this problem was to design our own controllers, since the motors themselves did not cause any problems.

After several tests of the designed motor controllers it appeared that it would extremely hard to control the motors on low speeds in the absence of feedback from the motors. It would therefore be necessary to measure the position of the rotor. There where two solutions to that problem: measuring back EMF and adding HAL sensors.

Instead of powering all three coils while the motor spins, only two coils could be powered while the third could be used to measure the back EMF (electro magnetic force). By measuring the back EMF the position of the rotor could be calculated. Although this seemed the easiest solution it appeared that the motor would produce too little back EMF at low speeds. And this is exactly where we would

need it the most.

This would only leave one other solution with the current motors and that is adding additional HAL sensors. These sensors would have the same function as the third coil measure mentioned in the previous paragraph: to get an EMF-measurement. If the EMF at a given position is known, the position of the rotor can be determined. If multiple sensors were added, even the direction of the rotor could be determined.

Fortunately there was already a small slot where the sensors could be placed so no additional adjustments would have to be made to the motors (see Figure 0.2).

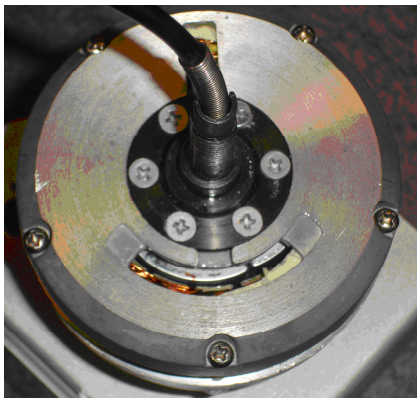

Figure 0.2 opened BLDC motor with slot free for HAL sensors

The HAL sensor where arranged in such a manor that the position, speed and direction could be determined and was capable to fit in the available space (Figure 0.3 and Figure 0.4).

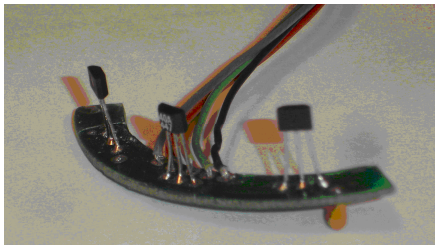

Figure 0.3 HAL sensor print

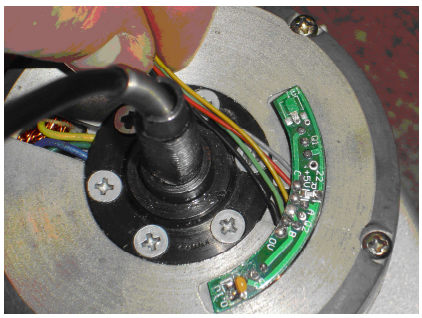

Figure 0.4 HAL sensors placed in BLDC Motor

Although the solution was almost finished, there was too little time left to thoroughly test the robot with the new controllers. It was therefore decided that the old controllers would be used instead.

Mechanics
There were only two minor problems with the frame. These where that the batteries where positioned horizontally? and this caused the batteries to fail after a certain amount of time. The other problem was the vibrations; there was no shock absorption of any kind in the robot. This caused the PC to reset on large shocks.

The problem of the batteries was easily adjusted by in creasing the space. The shock resistance was dealt with by placing the upper half free from the lower chassis by adding chock absorbers.

The chosen shock absorbers were the standard RC car chock absorbers, although after some testing it proved that these where of little use. Since no technical specifications were available, the influence of the shock absorbers on the system could not be calculated in advance.

Software
The software content only underwent some minor adjustments. Most work done on the software was to break down the old software and rebuild it. During this process the code was better documented and un-used functions such as those necessary for the touch sensors were removed.

Also the code was completely rearranged and structured for better readability and debugging purposes.

Vision
One of the largest problems during the event at 2008 was probably the lack of a functional vision system. Due to an invalid license the entire vision application was unable to operate. To prevent this problem, this year we looked into several other possible options. A few of these options where: an open standard library (openCV), Matlab (simulink) and Roborealm.

All these methods where analyzed and several test where performed. The option initially decide upon was Matlab, because there was a lot of information available on how to use it and a large number of standard libraries could be used. On top of the libraries Matlab also provided the option to generate C code that could be used in the existing main program. This would be a great advantage since the main program did not have to be rewritten, which would save a large amount of time.

Unfortunately in the last stage of the project it appeared that the program would not function on the embedded PC. Adjusting the program would consume too much time, hence another solution was chosen. Instead of making a complete program it was decided that an open source program, Roborealm, would be used.

The only problem with this program was that there was no Linux version. A windows emulator was therefore used instead. The great advantage of using this program is that a good functional program can be written in a short amount of time. The disadvantage is there is no in-depth understanding of the program.

Another disadvantage seems to be that some desired settings are difficult to make such as automatic light correction.

Robot Brothers EasyWheels and ReD in Field Robot Event 2009

Teemu Kemppainen, Teemu Koski, Jaakko Hirvelä, Johan Lillhannus, Tomi Turunen, Jussi Lehto, Ville Koivisto, Marko Niskanen and advisors Timo Oksanen, Jari Kostamo, Petro Tamminen
Helsinki University of Technology (TKK), Department of Automation and Systems Technology
Department of Mechanical Engineering
University of Helsinki, Department of Agrotechnology
contact: Otaniementie 17, 02150 Espoo, Finland. timo.oksanen@tkk.fi

Abstract

Two robots were built to Field Robot Event 2009. The first robot, EasyWheels, was designed to carry out all the tasks and the second one, ReD, was created mainly for freestyle purposes, but it was also supposed to take part in Task1. The robots were built by students from Helsinki University of Technology and University of Helsinki. EasyWheels won the second prize in the Field Robot Event 2009 and took all in all four prizes.

Introduction

In Field Robot Event 2009 the tasks were: 1) basic navigation between curved maize rows, 2) advanced navigation in straight maize rows with missing plants and with turnings based on program, 3) detecting weeds (green golf balls) and spraying those, and 4) freestyle where each team had to demonstrate what the robot could do in real world.

Figure 5: EasyWheels and ReD

EasyWheels (Figure 5) is a continuum from earlier Finnish robots that participated in the Field Robot Event (Honkanen et al 2005, Telama et al 2006, Maksimow et al 2007, Backman et al 2008). However, EasyWheels is completely redesigned and remade from scratch. No parts or software was used from the previous robots. Some of the EasyWheels' ideas are adopted from the last year's robot (Backman et al 2008) but this time the main idea in designing EasyWheels was to make it as

modular as possible.

The novel properties in the robot are: modular structure, advanced suspension system, two way driving, symmetric design, embedded computing with real time operating system, 3D CAD based design and CAM based manufacturing.

EasyWheels

Mechanics

The mechanics of the robot can be divided into four different areas as displayed in Figure 6. The area inside the red ellipse is the plate where the majority of the electric and electronic components are located. The frame and our advanced suspension system can be seen inside the blue circle and inside the yellow circle is the axel module. The cover which was designed and manufactured by our sponsor Valtra can be seen inside the green circle.

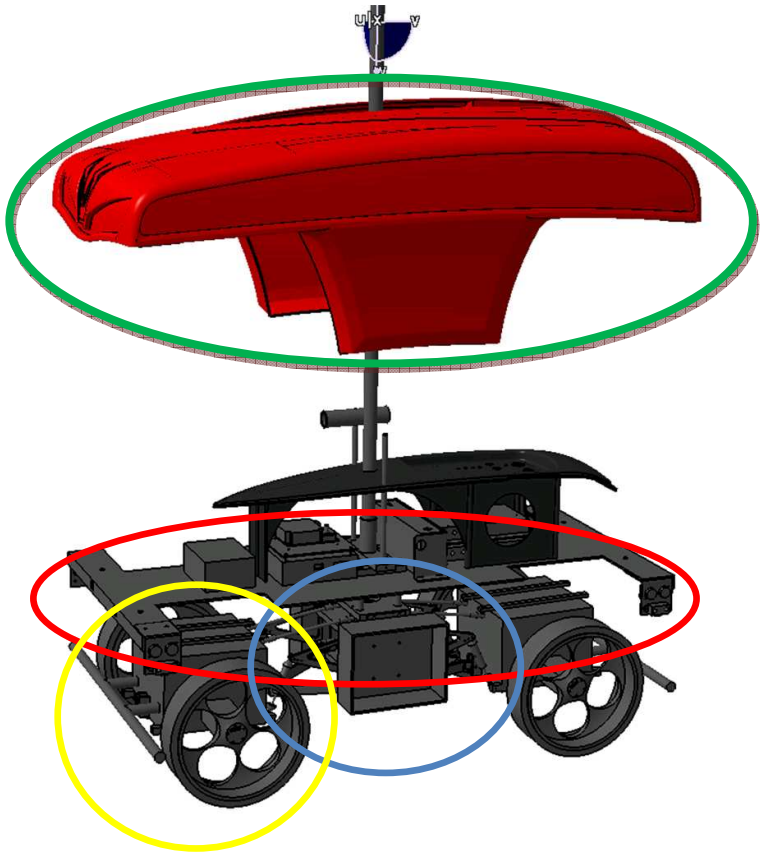

Figure 6: EasyWheels uncovered

Axel modules

The robot is designed and constructed of different modules. The modular design enables many ways for varying the construction and modular design could be adjusted for different tasks. This is an advantage in agriculture because the tasks depend on many variables and tasks vary depending on for example the crop and geological location. It is also an advantage that you can always have a spare module ready and in a case of a malfunction the module can easily and quickly be changed and the robot can continue its task while reparations are being done to the broken module.

The axel modules were designed and modelled using 3D-CAD software. All the components used in the axel module were modelled to ensure the parts would fit in a compact package. The model of the axel module is shown in Figure 7.

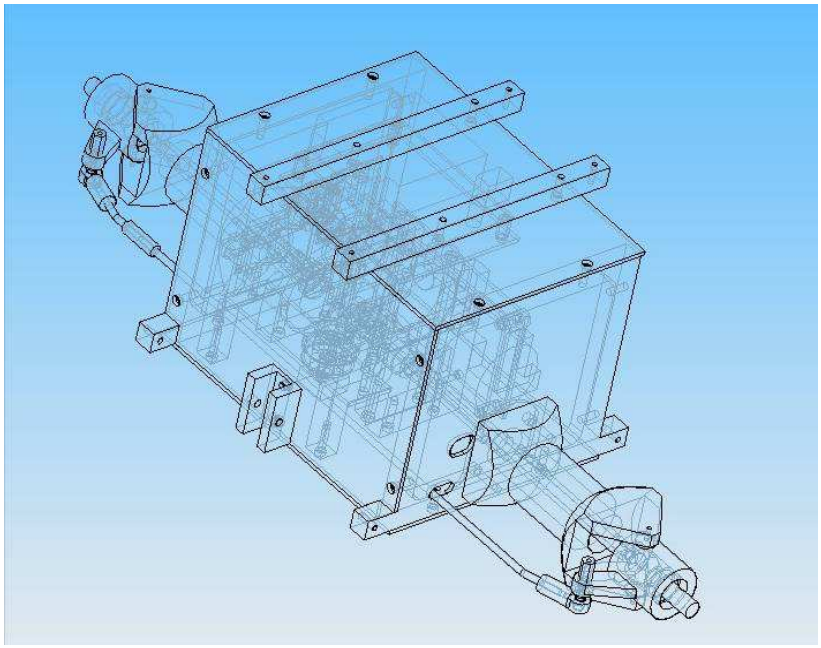

Figure 7: 3D CAD model of the axel module.

The main idea in the axel module was to include all the necessary parts needed for steering and driving the robot. The main components are: a) electric motor, b) reduction gears, c) differential, d) optical rotary encoder, e) steering servos including feedback, f) motor controller and g) microcontroller. These components were fitted into one box and the only interface to the module was a 12V power cable and a communication bus. The microcontroller in the module was designed to do the required feedback control inside the module. In the early stage different communication channels were considered but at the end RS-232 was chosen. It can be seen a picture of a complete axel module In Figure 8 and in Figure 9, the mechanically important parts are visualized and encircled.

Three identical axel modules were built: two axle modules were used in the robot and one was for spare (which proved to be a very good idea). As the team suffered from failures of the driving motors the spare module made it possible to replace the damaged module with a repaired one and the field tests could be continued with only short delays. While the field tests went on, the broken module was repaired by the other team members.

Figure 8: The axel module (with bumper bar attached)

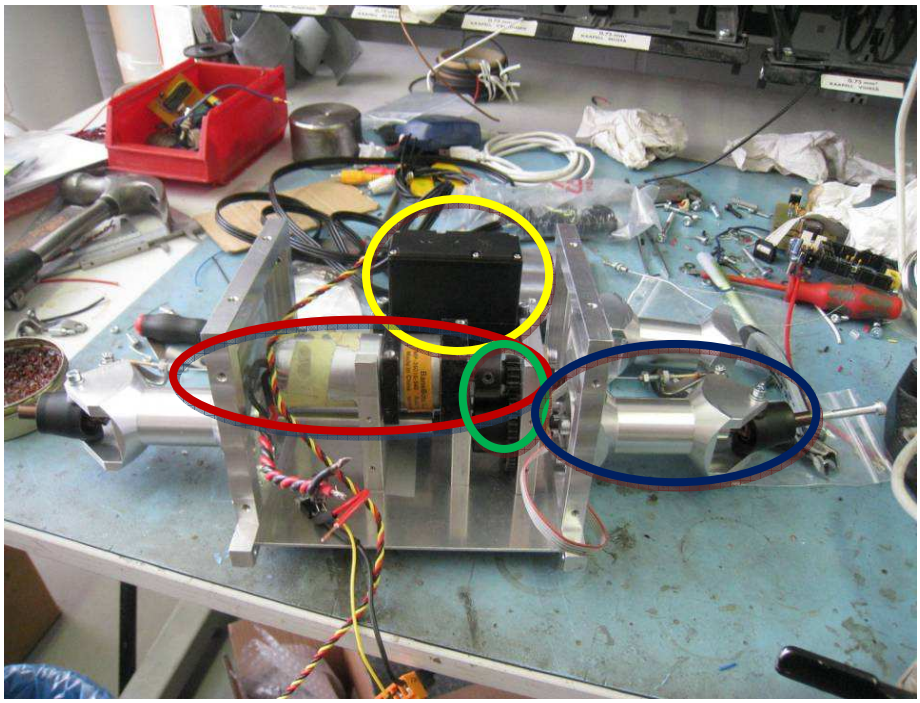

Figure 9: Main parts in axel module

In Figure 9, a Hitec HS-805 BB servo is shown in the yellow circle. The servo uses a wire system to turn the angle of the wheels. The wire starts from one steering block, circulates around the pulley of the servo and finally goes to the other steering block. The diameter of the pulley is optimised so that the full turning range of the servo is utilized and the servo delivers maximum torque to steering blocks. In order to improve the controllability of the steering angle, an external potentiometer was installed below the pulley wheel of the servo. The measurement of the steering angle was used to improve the precision of the steering angle and this was done by implementing a PI controller for the steering angle in the microcontroller. It was noticed that external measurement was required to compensate for the steady state error of the selected HS-805BB+ servo and adding a PI controller

improved the precision of steering significantly.

The red circle shows a Mabuchi RS-540 motor with a 16:1 transmission (Banebots) that deliver the torque to the wheels. Inside the green circle is the differential and inside the blue circle is the anchor and steering block.

The tires were cut from a Monstertruck 128/210-4.5 WhiteSpot tires and the wheels are CNC manufactured from a plastic bar. Most of the sheet metal parts seen in Figure 8 and Figure 9 were manufactured by our sponsor Laserle Oy and the remaining parts were manufactured at TKK. Laserle offered to cut all sheet metal parts for our robot according to our CAD drawings.

Frame and suspension system

The main idea for the design of the frame was adopted from the previous robot - 4M (Backman et al 2008). A four-wheel-driven and four-wheel-steered chassis was considered to be very functional as it was wanted to make sure the robot wouldn't get stuck on the field. The traction was improved by a balancing mechanism which ensured that each wheel applied the same contact force to the soil. In EasyWheels and 4M this was solved with a lever mechanism allowing the tires to adapt to the shapes of the surface. In addition to the balancing mechanism, a traditional suspension system was included in the frame of the robot. The 3D model of the chassis of the EasyWheels robot is shown in Figure 10.

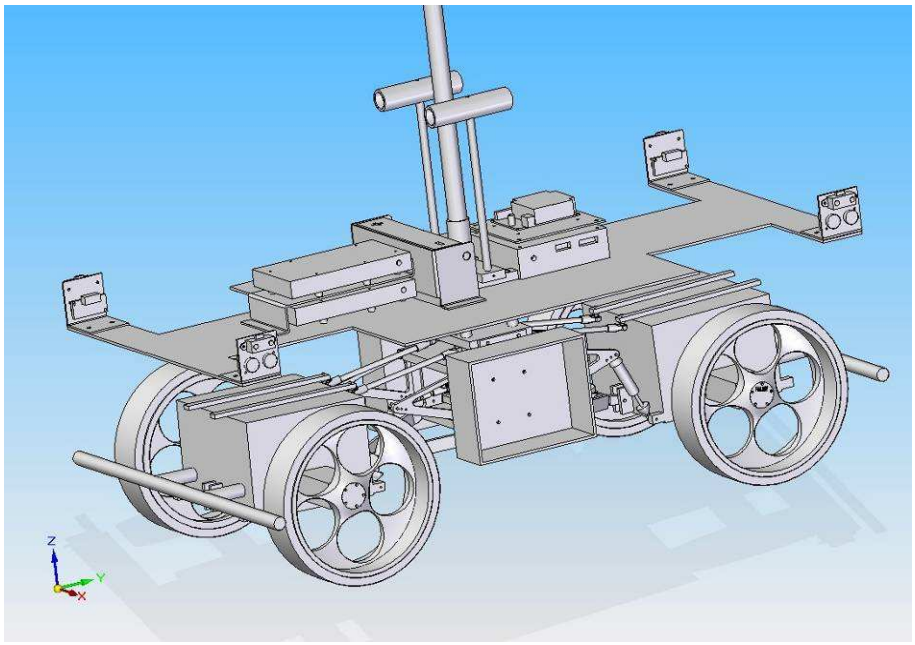

Figure 10: The chassis of the EasyWheels robot.

The idea was pretty much the same as last yeas but now that we had engines in the axel modules and not in the centre we needed to make some changes. One of them was to move the fulcrum from the top of the module to the side of the module. In that way it was also possible to relocate the part that controls the adapting in the lower frame (where 4M used to have its motor). The other reason was that EasyWheels was to be bigger than 4M.

The main idea was to connect all the shocks to a twisting plate that would rotate and guide all the four wheels to adjust the surface. Let's say that left front wheel would drop in a hole. Normally this would mean that left front tire would have less load on it as well as the right rear tire. In a worse case scenario this would result to the robot being immobilized because the differential gear would direct all the thrust to the wheels that are in the air.

In our solution the idea is to even the load on the tires so that every one of them would also have the same thrust from the engines. If the left front tire of our robot would drop in a hole it would drop all the way to the bottom of the hole, because as the left front tire would be dropping, so would the right rear tire. At the same time the right front tire and left rear tire would rise towards the frame.

As it is bigger than its predecessor, it also was heavier even though a full sized laptop was not used onboard as was in 4M. That resulted in some trouble with the suspension. The original shocks that came with the shock absorber were obviously too light. They were designed for a 1.5 kg radio controlled car. With some searching and luck we did find suitable springs for the robot. The main parts of the frame are milled or laser cut aluminium slabs and they are connected simply with screws. The most important rotating parts were attached with ball bearings to be sure that they operate as planned.

Plate

The idea was to have all the electronic systems (excluding those in axel modules) in one plate. The plate was designed to have enough room for systems. Despite of that requirement, at the end it was considered too small. One reason for that was that the plate had to be shrink down in order to make the cover to wrap also the batteries. Shrinking down included also some designed bends that was done to increase the plate's stiffness; though the bends were removed the stiffness of the assembly plate was still sufficient. The space required for cables was underestimated. After all modifications we had less space for electronics and wirings got messier than planned.

Cover

The cover for the robot was designed by Valtra and they have manufactured it using a 3D plastic printer. This was part of the sponsor agreement; the robot should not look like 80's video cassette recorder. Despite of technical limitations for designer, like free space in front of axel modules for tools, the design is very nice and functional for maintenance purposes. (Figure 11)

The cover was designed to be waterproof, it consists of two parts: one fixed part where control buttons are (in Figure 1, the black one), and the removable cover that allows easy access for maintenance. The local control buttons like for example start, stop, force turning and emergency stop were very handy when calibrations were made, because the robot did not always do what we expected it to do.

Figure 11: Cover design by Valtra.

Electronics

The electronics of EasyWheels can be split up into two areas: axel module electronics and plate

electronics.

At the beginning it was decided to use only one kind of microcontrollers in this project. As the proto board used also in 4M was reported to be a good, the same model was used also in EasyWheels. The proto board is Futurlec ATMEGA which incorporates AVR ATmega128 microcontroller. One controller is inside every axel module and one is used to read sensors in the plate.

The software development for AVR was done using CodeVisionAVR by HP InfoTech.

Axel module

The electronics in the axel module are all related to the steering and driving of the wheels. The system is built up around an ATMega128 micro controller/ Futurlec ATMEGA proto board. The components that are connected to the board are the Hitec HS-805 BB servo explained in chapter 1.1 and the potentiometer that works as a feedback for its position. For motor feedback a Avago Technologies optical encoder (500 PPR) was used. With the info obtained from the encoder (motor rpm and direction) the motors speed is set with either a PWM value or rpm value that is obtained through a PI controller. PWM signal is directed to custom a build PWM amplifier, the amplifier is based on the same design as the one used in 4M. The PWM amplifier is shown in Figure 12.

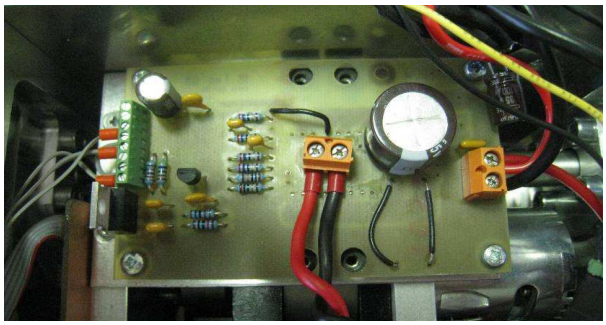

Figure 12: Motor PWM amplifier.

The controller is programmed so that it has 3 modes. If the module is in mode 0 no functions will work and the controllers are in standby. Mode 1 is for testing and as backup for normal drive. In mode 1 the servo and motor is given straight PWM values (open loop) so that it is easy to make tests. In mode 2 all the PI controllers start to work and this is the mode that the robot will have in the field. The microcontroller sends/receives information to/from the PC located at the plate.

Plate

The robot contains four ultra sonic sensors (SRF08) which are used for measuring distance from the robot to the corn plants in row. The measuring distance is set to about 60 cm and sampling time is 50ms. All the ultra sonic sensors are connected to I2c local network which is operated by Atmel 128 microcontroller. The ultrasonic sensors are programmed so that they first fire the front sensors and then read them one at the time and then the same procedure with the rear US sensors. Offset of 25ms between front and rear sensors and gain value optimisation is used to prevent false echoes. The same local net I2C contain also two 2D magnetometers that detects earth magnetic field in x and y directions. Sensors are installed in a 90 degree angle, so that measured data from the 2D sensors can be combined so that a 3D data can be obtained, it was cheaper to buy two 2D magnetometers than one 3D sensor.

The robot contains also four infrared distance sensors (SHARP GP2D12 4-20cm) which are located underneath the ultrasonic sensors and the practical measuring distance of the IR sensors is about 30 cm. The sensors are controlled by the same Atmel microcontroller that handles the ultrasonic's

and magnetometers.

Due to tight time scale and lack of space on the robot we had to cancel the planned gyro sensors and the inclinometer. The gyros and the inclinometer was tested and controlled by Atmel microcontroller and connection was made via SPI ports.

Machine vision was made with Logitech webcam that was a little bit modified and re boxed. Because the robot needed to drive in both directions the webcam was installed on top of a 360 degrees turning servomotor. The night before the race we decided to chance servo motor because reliability problems with turning accuracy. On the Field Robot Event the servomotor was an ordinary 180 degrees servo and due to lack of test time it did not work as expected.

Computers, microcontrollers and other electronic devices are getting their electricity from a lead acid battery that is located on the assembly plate (12V / 2.1A). Also microcontrollers inside the axel modules are getting their electricity from this battery. The battery can run about one hour before the voltage drops below critical. Battery powers one main computer, two additional computers which runs image processing and machine vision, three Atmel 128 microprocessors and other gadgets like WLAN and sensors.

Machine vision system

Machine vision system consists of two different parts: the system that detects golf balls (weed) and the system that detects the center of the maize rows as well as the beginning and the end of the maize rows. Due to computing problems it was decided to have one processor for each part.

Row detection

This system is divided into adaptive RGB-color channel altering, simple color thresholding, detection of the centerline and into calculating the output.

The idea of the RGB-color channel altering is that because thresholding is based simply on color, we need an algorithm that would keep the RGB-colors the same in different daylight conditions. First the histogram for each color is calculated. Then the beginning and the end of each RGB-color channels are found so that we first start from intensity value 0 and going "upwards" until we have found 10% of the color channel that we have been looking for. The same idea is applied when finding the end of the color channel.

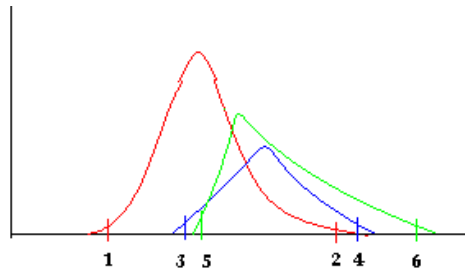

Figure 13: Histogram for each of the RGB-colors and their representative beginning and end points

In Figure 13 1 represents the beginning of the red-channel and 2 the end of the channel. Respectively 3 and 4 represent the blue-channel and 5 and 6 the green-channel. After this each of the color channels are expanded so that in a new RGB-color histogram each of the beginning points

are 0 of intensity and end points 255. The formula for calculating a new intensity value for each pixel is (this formula needs to be calculated for each color):

$$I(x,y) = \frac{I(x0,y0) - Cmin}{Cmax - Cmin} * 255,$$

where I(x,y) is the new intensity value, I(x0,y0) the current intensity value, Cmax the end point of the current color channel and Cmin the beginning point of the current color channel.

After these calculations the RGB-color histogram should look something like this:

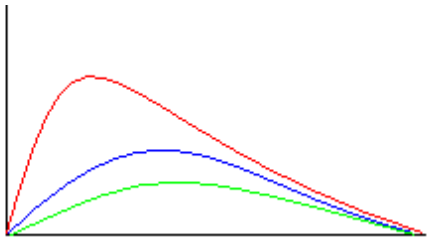

Figure 14: The histogram after RGB-color altering

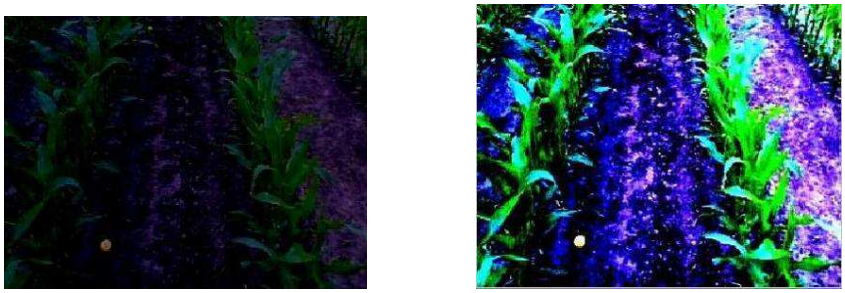

Figure 15: Original image (left), altered image (right)

However, this system was not used because it was too heavy.

The image is thresholded so that green vegetation can be seen from the thresholded image. This results in a black and white image, where white corresponds green colors and black other colors.

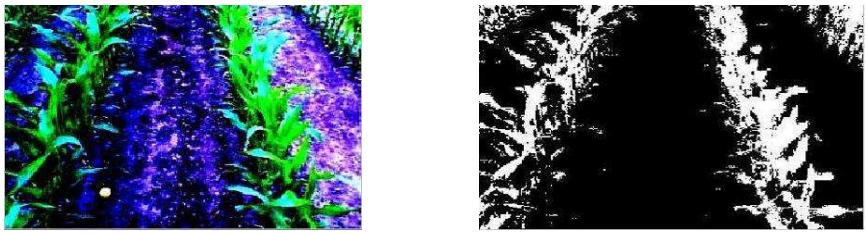

Figure 16: Altered image (left), thresholded image (right)

The used thresholding uses a following function:

$$f(x,y) = \begin{cases} 255, & if\ green\ channel > r + red\ channel\ and\ green\ channel > b + blue\ channel \\ 0, & else \end{cases}$$

where r and b are parameters which can be set.

After thresholding, the system tries locate the maize rows. The algorithm starts from the bottom center of the image and goes left and right until it has found enough white pixels in a 3x5 pixel region. Then a red circle is added to the resulting image. When it has found the maize rows from left and right it calculates the average of these x-coordinates and a new centerline point has been found. A blue circle represents this in the resulting image. When the whole image has been checked for maize rows, a straight line is plotted on the calculated centerpoints (green line). Blue line indicates the direction of the robot in Figure 17.

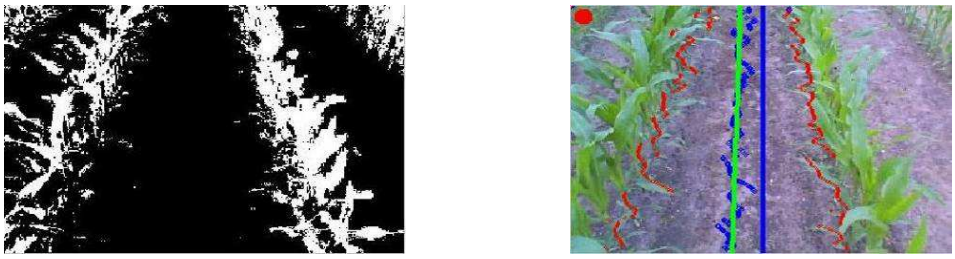

Figure 17: Thresholded image (left), resulting image (right)

Now the robot calculates the output using the robot's centerline and the calculated centerline from the analyzed image. Output consists of the distance between the robot's center and the calculated centerline and the angle (alfa) between these two lines.

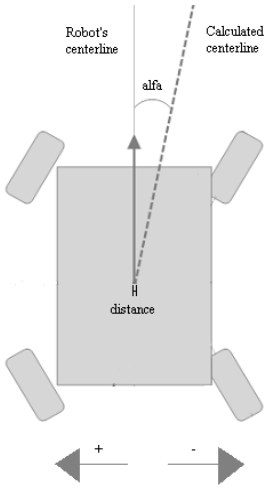

Figure 18: distance and alfa variables

Before the output can be calculated a few conversions have to be calculated. First the upper length

(d1a) and lower length (d2a) of the camera image have to be calculated. This is done by:

$$d1x = 2 * \frac{4}{3} * \tan\frac{a}{2} * \sqrt{h^2 * (dix)^2}$$

where h is the height of the camera from the ground and dix the distance between the center of the robot and the camera's upper or lower view.

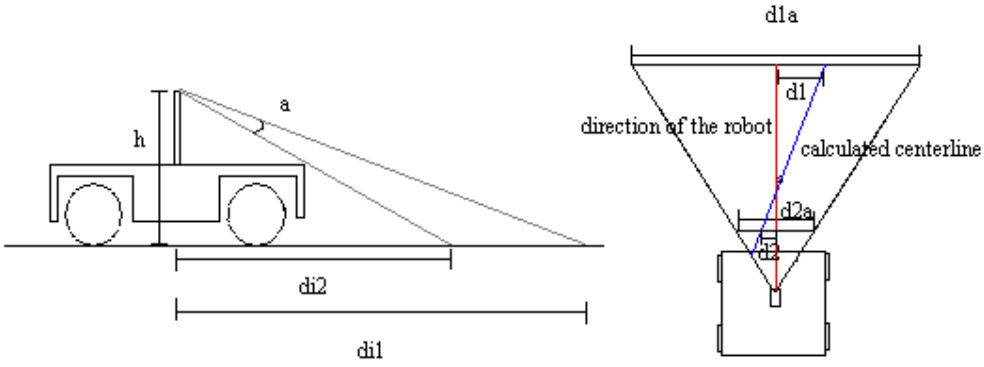

Figure 19

The alfa is then calculated as follows (radians):

$$alfa = \tan^{-1}\left(\frac{\frac{d1a * d1}{320} - \frac{d2a * d2}{320}}{di1 - di2}\right)$$

And the distance (meters):

$$distance = \frac{d2a * d2}{320} - di2 * \tan^{-1}(alfa)$$

Now the detected centerline of corn row is in metric scale and ready for sensor fusion and navigation.

Golf ball detection

This system is divided into multiband thresholding and hough transform. The thresholding works by assigning for each of the RGB-colors a lower and upper threshold value. The resulted image is black and white image where white pixels correspond golfball green.

Figure 20: Original image (left), thresholded image (right)

But the resulted image has “false” white pixels which can be removed by averaging. This is done by checking a 3x3 pixel area and if there are enough white pixels the intensity of the middle pixel is left to 255. If the criteria is not met, the intensity is changed to 0.

Figure 21: Smoothed image

After this we use canny edge detection to find the edges of the thresholded image.

Figure 22: Canny edge detection applied to smoothed image

The next step is to use Hough transform to find objects that are circle of shape. Basic idea is that we draw a circle of radius r pixels in every white pixel in a completely new image. This circle just adds the intensity value by 1 in the new image.

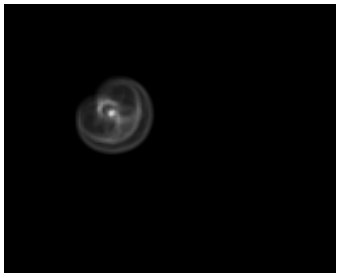

Figure 23: Hough transform applied to previous image

Next the algorithm scans the Hough transform picture and if a pixel has an intensity value greater than I (I can be set, but it was set to 200) the algorithm has found a ball and draws a circle in the resulting image.

Figure 24: Resulting image

After the system has found x times (x can be set, it was set to 3) a golf ball below a certain line, the system informs that the ball has now been passed. It gives 128 if the ball was on the left side of the image 1 if it was on the right 0 if there was no ball and 129 if there were balls on both side of the image.

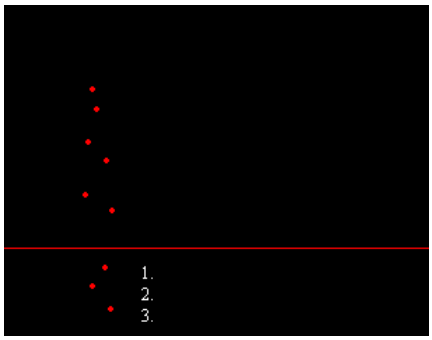

Figure 25: Image where the detected golfballs can be seen

Navigation

The navigation development started when the robot was still on the drawing board. Therefore a simulator was developed. The same ideas from previous robots (Backman et al 2008, Maksimow et al 2007, Telama et al 2006) was recirculated. The kinematic equations for 4 wheel steered robot were written and it was modelled using Simulink. All the distance sensor measurements (ultrasonic, infrared) were simulated by using simple beam angle collision test by polygon algorithm. The image of camera on top of mast was created by modelling the field using Matlab Virtual Reality toolbox and the rendered image was captured from screen. Simulators' graphical output is shown in Figure 26.

Machine vision measurements were done by first creating a VRML file from field data by adding 1 plant to each maize location in data file. A picture was then made for each simulating step using its

current location as input to Matlabs Virtual Reality toolbox. The picture from VRML block was then sent into MEX file, which is Matlabs way to run C-codes. This MEX file calls the machine vision codes that were the same as codes in real systems.

All the navigation algorithm are developed in Simulink and the controller part was converted to C-code using Simulink RTW. This was very useful, first the same algorithm can be developed, tested and tuned using simulator, and then the same algorithm can be run in the robots real time system.

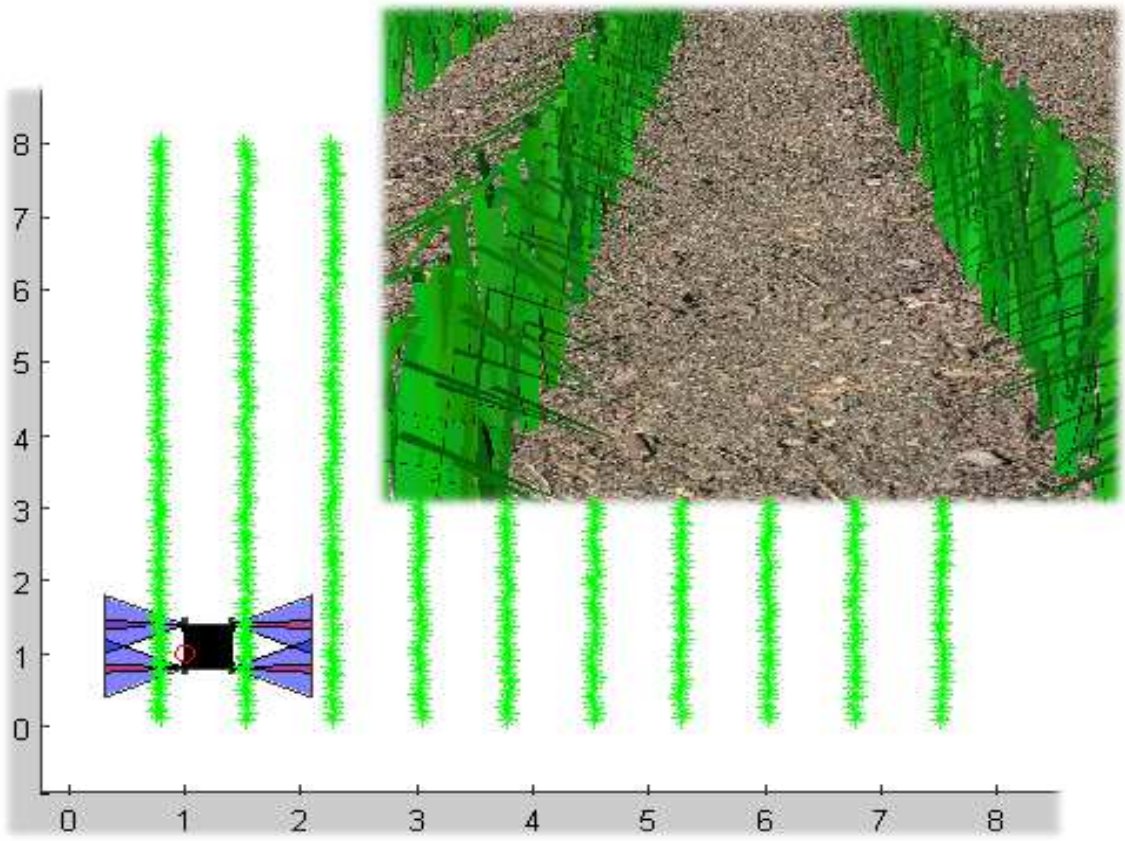

Figure 26: Simulator graphical outputs

The controller consists of blocks of which only one was active at the time and modes were used to control which block was in use at which time. Each task was divided into individual blocks such as normal driving in middle of row with row changes always to next row. For staying in the middle of the row, two different controllers were made: simple one and advanced navigation.

Row end operations

Row changing was implemented using Simulink Stateflow and two different state charts were done. The first for always changing to next row using "crab turns" (Figure 27) and the second for following patterns which could skip rows when changing rows. Crab turn was defined by previous teams and it means that first the robot turns all wheels pointing to one angle (like a crab does), drives until travelled equal to half the row width, then stops, turns all the wheel to opposite angle and drives back without actually turning the robot.

For advanced pattern turns distances were used, which were calculated from control history. When

state changes to out from row, the robot turns 90 degrees (which is measured by compass) after this the robot drives straight the distance that is calculated by width of row multiplied with the amount of turns need to be skipped, and reducing an amount of two times turning radius (equation 2.4.1). Also if turn amount is 1, "crab turn" is performed and if next row is same as earlier just the opposite direction, the robot drives out from row turns the camera around and then goes backward in the same row. Due to limited test time pattern turns couldn't be made working although pattern turning did work in simulator. During competition only "crab turns" worked well enough on the real system.

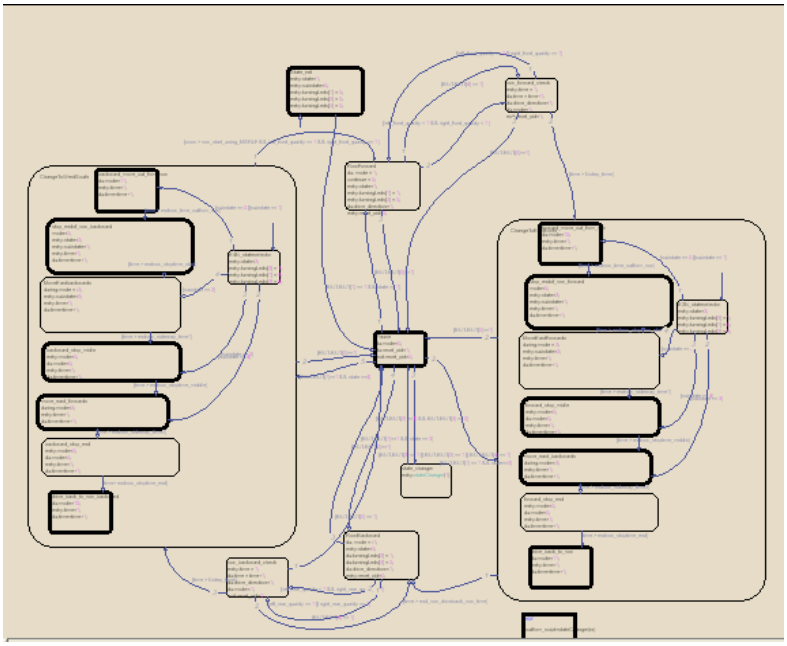

Figure 27: Simple row changing flow chart for doing "crap turn"

$$dist = (next\_row\_number \bullet row\_width) - 2 \bullet turning\_radius \qquad (2.4.1)$$

Advanced and simple navigation

Simple control strategy locked current rear wheels into middle position and uses front distance measurements difference to steer the front wheels by using a PID controller. After performing "crab turn" rear and front side are changed in scripts and the robot just locks its new rear wheels into middle position and start steering with its new front wheels using earlier rear ultrasound measurements which are now its front measurements.

Advanced mode consisted of weighted least square method that fits 1 line for latest 20 measurements from ultra sound-and infra red sensors for each side of robot. The locations of those measurements were placed into coordinates of its current location and calculating back to its previous location for each measurements using robots 20 latest controls. Outputs from this line fitting are one angle, distances to rows in robots left and right side and goodness number of current fit. These calculations were then combined with machine vision outputs by using weights and goodness number. Result from these combination operations are difference from middle and angle difference from going straight. Combined angle- and distance difference were inputs of 2 PIDs, the first controls angle difference trying to set robot straight and the second one tries to keep the robot going middle of row. Both controllers operated as PI controllers because the derivative term made it unstable in simulations. Final wheel angles for front and rear axel modules were calculated

using these controllers' outputs and simplified odometer model.

Real system implementation and log file simulator

Both control systems worked in simulator but due to lack of testing time advanced mode wasn't robust enough for using it in competition and only simplified control strategy was used. In both cases row ending was decided using two conditions: machine vision parameter that was "robot in row"; and 10 latest front side ultrasound measurements if all those were out of boundaries. Although there was a possibility of not using machine vision at all and only drive with measurements from ultra sound sensors, by only turning weight parameter to 0.

The system that was used in the real robot also included a front- and a rear block as shown in Figure 28 this changed the robots inputs which in most cases are pulses to engineering units and rear controllers turns controllers outputs back into corresponding pulse amounts. To be able to use this in the real robot it was generated into C-code by using Matlabs RTW C-code generator. In Matlab 2007b some problems with C-code generation occured. This caused lots of unnecessary problems such that it forgot some parameters from the C-code that were supposed to be changed in GUI and code optimization rounded some values without informing the user. Also Matlab 2007b seemed to have some sort of stability problems and crashed every now and then.

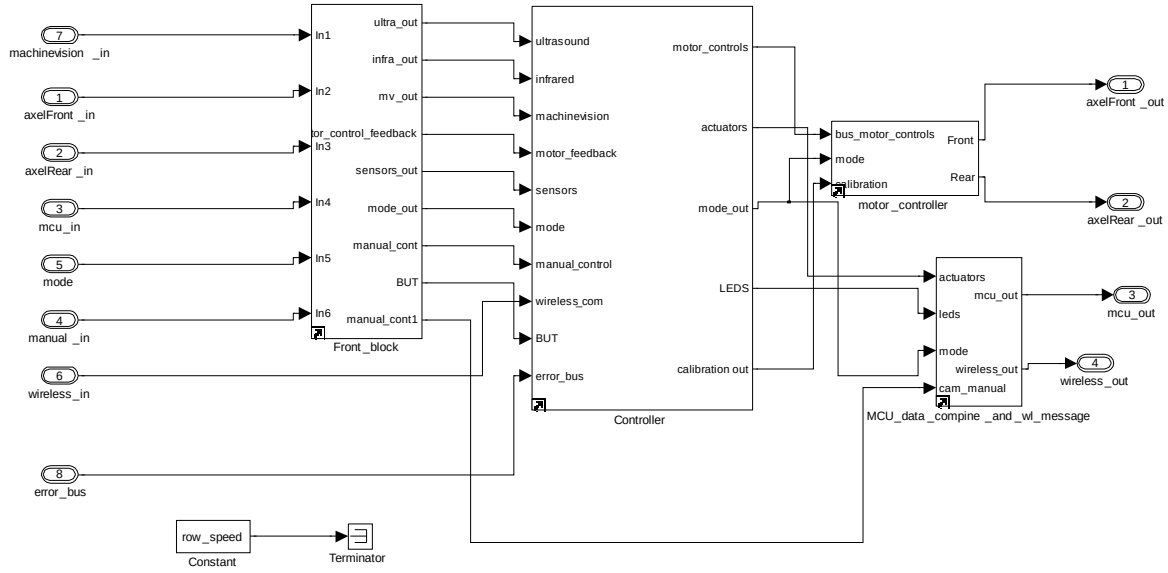

Figure 28: Simulink model of real robot control system, front and rear blocks

The simulator that was used to calculate controller states from log files was done in a hurry and it had some performance problems which occurred when the log file size increased and requirements for physical memory increased significantly in long log files.

In the controller part many things could be been done better like not using multiple copies of the same element in different task modes by only doing rearrangement of blocks and their tasks. Also benefits of making libraries were understood by the controllers creator too late to take full benefits from these.

Computing system

In contrary to earlier robots, EasyWheels uses embedded distributed computing instead of an on-board laptop. Earlier robots couldn't reach strict real time, because of the non-real time operating system, Windows XP. EasyWheels' computers use Microsoft Windows CE 6.0, which is a real time

operating system. With non-real time operating systems the problem usually was non-deterministic operating system's background processes that could freeze or interrupt critical navigation and machine vision algorithms. With Windows CE this kind of behaviour should not happen. In addition to real time operating system EasyWheels' computers are embedded. The goal was to achieve a modular computing platform with strict real time capabilities.

Hardware

EasyWheels has three embedded computers all running Windows CE 6.0. Two of the computers are for machine vision and one is for navigation. Machine vision computers are Toradex Colibri PXA320 embedded computers on Toradex Protea carrier board (Figure 29). The navigation computer is ICOP's eBox-4300 (Figure 30).

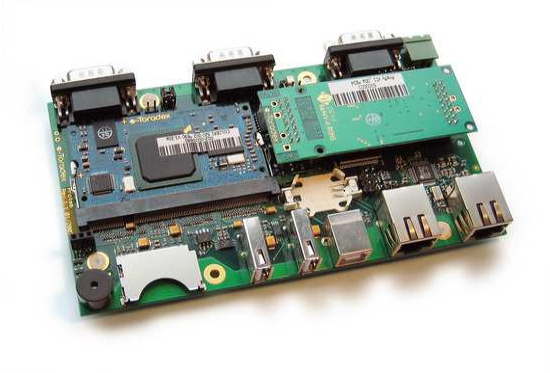

Figure 29: Toradex Protea carrier board with Colibri module (Toradex)

Figure 30: Ebox 4300 (EmbeddedPC.NET)

Colibri PXA320 has a very low power consumption with a great amount of computing power. Colibri has an ARM processor running at 806 MHz, but no floating-point unit (FPU). It was selected to be used for machine vision because of its computing power, the FPU was considered as an acceptable lack, as machine vision doesn't necessarily need floating-point computing as RGB images consists of 8bit integers. Naturally floating-point operations are possible, but through emulation they are very slow. One Colibri was used for row detection and the other for golf ball detection. As a navigation computer eBox is very efficient. EBox uses conventional x86 PC hardware and therefore has a FPU. With VIA Eden ULV 500 MHz processor eBox can easily run complex navigation algorithms which can't be run on a Colibri. The three computers are networked via Ethernet, using a WLAN router. With networked design none of the computers (or algorithms) can freeze or slow down overall computing.

Of course one of the criteria was also a small size and both computers achieves this. Protea carrier board's dimensions are 86.5 mm x 117.4 mm with around 15 mm thickness. Ebox's dimensions are 115 mm x 115 mm x 35 mm.

Software

Modular design was also a goal with the software development. All code is self written in C++ except

for the Matlab Real-Time Workshop generated navigation algorithms.

Machine vision software is designed to be as modular as possible. Even a less experienced developer can create a usable machine vision algorithm with the design. The machine vision itself is separated into several blocks, each computing a small part of the overall algorithms work, for example one block could be colour conversion, one threshold and so on. The algorithm is a "script" in a C++ -header so that the machine vision developer only needs to create necessary blocks and connect them in the script. Blocks are naturally designed to be reusable and as generic as possible.

Navigation software uses the same modular idea that machine vision, but it has no script or blocks. The machine vision algorithm itself is Matlab Simulink RTW generated code with C interface. Modularity is achieved so that navigation software is independent from the generated code. Only a small part of the interface needs to be rewritten if the algorithm changes drastically. If none of the inputs, outputs or tunable parameters change as the algorithm changes, no editing is needed for the software.

All of the software provides server and client interfaces. Every computer can be used and tested alone or together (Figure 31). Computers talk to each other via Ethernet, remote user interface is via WLAN.

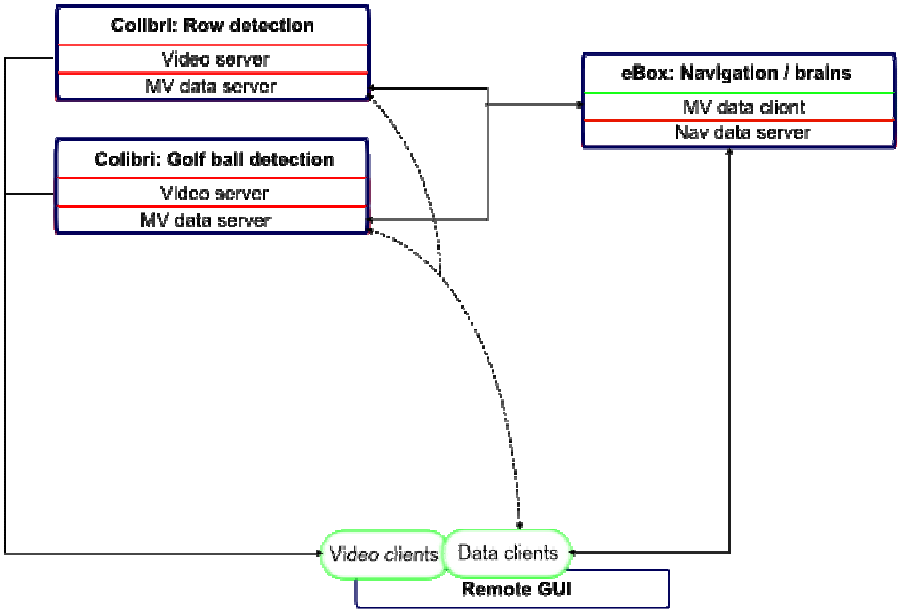

Figure 31: Software interfaces

Even though the interface is quite simple and easy to use for the algorithm developer, a huge amount of lower level C++ code had to be written. Most of the unexpected extra work was due to distributed computing and Windows CE. As none of the previous robots used distributed nor embedded computing, no one had any previous experience on the area. Distribution of computing power itself was expected to cause some extra work, but no one could predict the amount of headache Windows CE caused. Many parts of the code had to be rewritten several times as more and more "features" of Windows CE was revealed. Some of the features could have been avoided with previous experience with Windows CE. The biggest avoidable problem with CE was the lack of RTTI (Run-Time Type Information). This effectively prevented us from using Boost C++ libraries at all. The original idea was to use Boost.Asio with networking. After that deficiency MFC (Microsoft Foundation Class) was considered to be the choice for networking, but it was so buggy that it didn't work correctly on CE. Finally all networking code was written with low-level Winsock 1.1. This forced

us also to write all the serialization code from scratch.

It turned out that Windows CE lacks all the advanced features (available e.g. in Win32 environment) that enable the developer to write sophisticated code. Poor documentation of Microsoft's code leaded to finding out all the features the hard way.

For machine vision OpenCV main libraries was ported to Windows CE, only some minor changes had to be done. This was considered a major breakthrough to use it in Colibri. Unfortunately it was learned that OpenCV utilizes floating point computing and typing internally even if the image type was integer. The result was that it was practically impossible to use any advanced OpenCV functions in real time computing, as the code had to be optimized all around.

ReD

As there was a considerable difference in knowledge of robotics, electronics and software within the group in the beginning of this robot project, it was decided that the students at the University of Helsinki should get acquainted with basic robotics first. The requirement was that a monster truck (Tamiya TXT-1) should be converted to a simple field robot being able to do the basic task of the competition indoors (driving between rows and making turnings to the next row). The idea was to make this with only two ultrasonic rangers, one steering servo, one speed feedback sensor and a AVR microcontroller.

The goal was to build the basic robot during autumn semester and the co-operation with other robot and "the tool" during spring semester. "The tool" had to be some real application in fields, and for the freestyle it was decided that two robots do interactive mechanical weed control, and this inspired the name of ReD (Remote Destroyer).

Mechanics

ReD is based on Tamiya TXT-1 RC-car chassis (Figure 32). The chassis has advanced suspension for both axles. Transmission consists of two Graupner Speed 500 motors mounted parallel in main gearbox, where also power is divaded for two cardan shafts, one for each axle. Front and rear axles are identical. Axles have differential gears. ReD has continuous 4-wheel drive, and as both axles are identical, it has also 4-wheel steering. In row it uses only front axle steering, but in headland it steers with both axles to improve turning.

Figure 32: ReD

Electronics

ReD uses three voltage levels, +12V, +6V and +5V, which all share the common negative ground. Two 6V NiMh batteries are run in series to deliver the +12V main power to the “brains” of ReD, Futurlec ATMEGA microcontroller. +12V main power is also used by the self built DC-motor speed controller unit which controls the two Graupner Speed 500 DC-motors, CMUCAM3 machine vision camera and by the +5V regulator circuit on the self made power distributor board. Regulated +5V from this self made board is used by the Xbee Series 2 wireless serial communication device which is used for wireless serial communication between the ReD and EasyWheels. +5V output voltage directly from the ATMega AVR128 is used by two SRF04 ultra sonic sensors which are used for the navigation in the row, Honeywell hall-sensor which are used for odometry and for measuring the drive speed and also by CMPS03 digital compass which is used for turn angle detection at the end of the row. Two Acoms AS18-MG steering servos utilize +6V input voltage which is taken separately from one of the main batteries.

User interface includes three power switches: the main power switch and two separate switches for the AtMega 128AVR and CMUCAM3. User interface also has two push-buttons, one for compass calibration and one for switching the turn direction at the end of the row. Five LEDs on a custom made circuit board are also included in the user interface as indicators for different events. Two red LEDs show the current turn direction setting to be used at the end of the row, a green led indicates the power-on for the ATMega AVR128, one yellow led indicates the initiation of the starting sequence and one red led is reserved for error situations.

Machine vision

Machine vision system in the ReD is used to measure the distance between the ReD and EasyWheels robot, in freestyle. This distance information is used to calculate when the weed destroying unit has to be switched on and off. When information from the EasyWheels arrives about the location of the weed it has detected in the row, the distance between the ReD and EasyWheels is marked and based on to that distance information the weed destroying unit is switched on and off at appropriate time.

The base for the machine vision system is a CMUCAM3 camera unit which comes with the ready algorithms for image color detection. Ready algorithms are mainly used for image color detection, only slight modifications are made. The algorithm which is used divides the image to rows with the height of 1px and then processes each row separately while trying to find the specified color. After all the rows have been processed the information gathered during the process is assembled and information where the specified color in the image exists is shown. Machine vision system is utilized so that ReD is able to follow the EasyWheels robot and calculate the distance between the two robots. Basically the algorithm detects two red light sources (LED spot lights on the EasyWheels) from the image. After the detection the distance between the two objects is calculated. When this distance information is placed on to the pre-calculated formula, the distance between the two light sources and the camera can be calculated.

Navigation in row

Navigation in row is based on two SRF04 ultrasonic rangers. Measurements over 500 mm are filtered away to ensure that reflection from wrong rows won't be measured (e.g. if there are plants missing). If both measurements are usable, effective inter row spacing is calculated and recorded for further use. If only the other measurement is usable, hypothetical location of the other row is calculated based on recorded effective inter row spacing. This row estimation, as we call it, enables robot to follow only one wall. Default inter row spacing can be set in program. Distance between middle point of robot and middle point of the row is calculated either from both measurements (if both are usable), or from one measurement and calculated location of the other row. This distance is fed into the PID-controller that calculates new PWM-signal for steering servo. If both measurements are not usable during 500 mm driven distance, the robot starts headland turning. This distance can also be set in program.

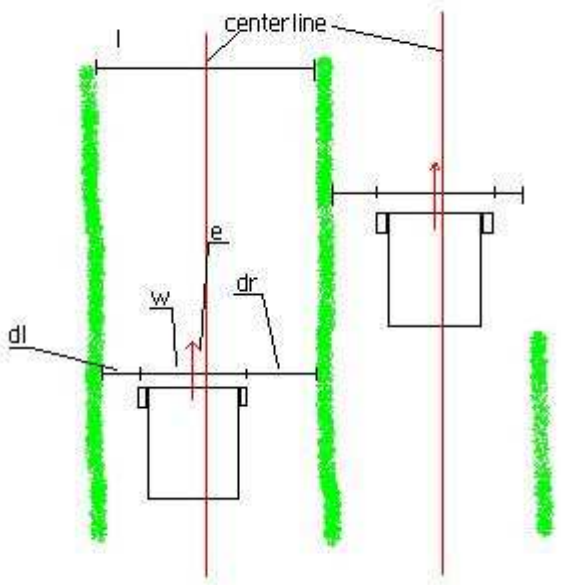

Figure 33: Navigation in row

In Figure 33 l = inter row spacing, w = distance between ultrasonic rangers, dl = left distance to row, dr = right distance to row, e = error (distance between centerlines of row and robot).

Error is calculated as follows: e = dl – dr

and missing measurement: dr = l – w – dl

Computing system

Programming is done with FlowCode flowchart programming environment (Figure 34), which generates C-code. All routines mentioned in this chapter are done in FlowCode macros (macros are generated as functions in C-code). Macros are used as much as possible, e.g. PID-controller, row estimations and headland turnings are in their own macros. This makes flowchart very easy to read. Interrupt routines are the main core of the program. Timer interrupts are used for pulse generation for servos and motor speed controller and one timer interrupt is used to measure approximately 16 ms time period to schedule trigging of ultrasonic rangers and data transmitting to EasyWheels. External interrupts are caused by hall-sensor, ultrasonic rangers and a push button for switching next headland turning direction. Data receiving causes USART-interrupt when the camera or EasyWheels sends data to ReD. Only very short routines are executed in interrupt macros. Longer routines are flagged to be executed in main loop right after interrupt. Some routines couldn't be done with FlowCode's flowchart, but it's possible to add C-codes into flowchart.

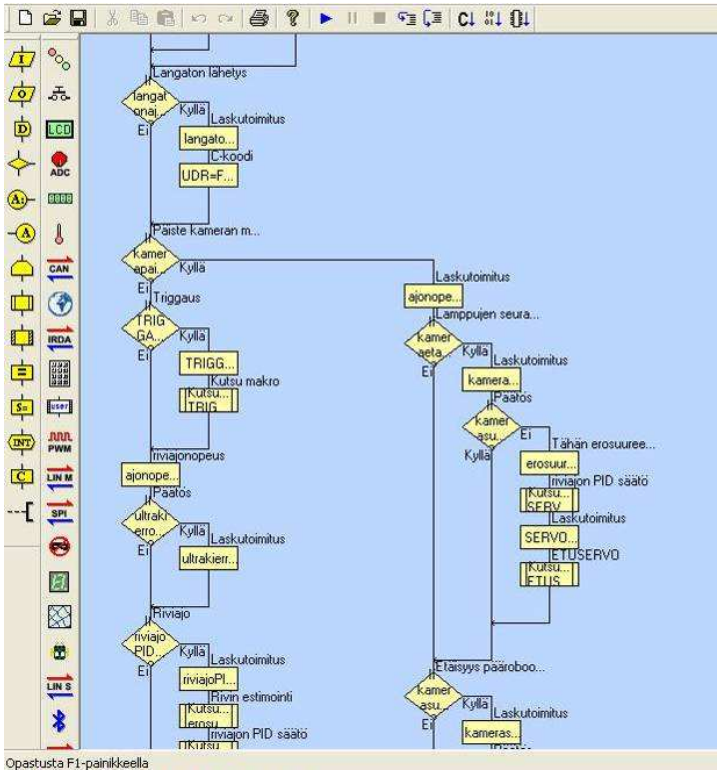

Figure 34: FlowCode

Destroyer unit

Destroying unit consists of simple chassis and two Mabuchi 540 motors. When EasyWheels tells ReD, that there is a weed in left or in right, camera is used to measure distance between ReD and EasyWheels. ReD measures distance to place where weed was found with hall-sensor. When weed is reached, ReD starts up its destroying motor in left or right depending on which side weed is found. Motors are controlled with transistors and relays.

Freestyle

In freestyle task, both EasyWheels and ReD would have worked together. Unfortunately ReD had problems with electronics, and this freestyle had to be skipped. The following will explain how it would have worked.

Idea

The idea of freestyle was to destroy weeds between rows. Shortly, EasyWheels was meant to detect weeds (golfballs), and tell ReD where to turn on destroying its (Figure 35). It was meant to use machine vision as a virtual drawbar so that ReD would have followed EasyWheels right behind. Data transmission would have been wireless. Machine vision and data transmission as well as destroying itself are described in chapters below.

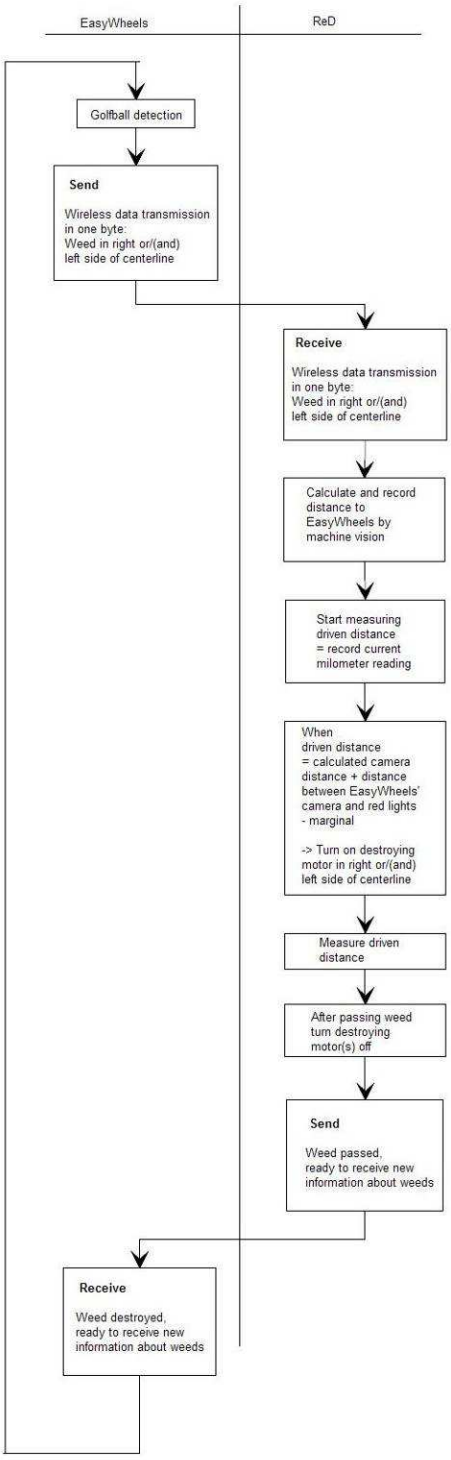

Figure 35: Simplified flowchart of planned freestyle

Virtual drawbar

Machine vision would have been used to make ReD follow EasyWheels. The idea of the virtual drawbar is simple: The rear end of EasyWheels was equipped with two red lights. In the front of ReD is a camera which would have been used to detect those two lights. As the distance between the two lights was known, the distance from ReD's camera to EasyWheels rear lights could have been calculated, as those distances are approximately proportional (Figure 35).

Destroying

When ReD would have been told by EasyWheels, that there was weed in either right or left or both side of centerline, ReD would have taken care of destroying incident during the rest of time. At the moment when information about weed would have reached ReD, it would have calculated distance from EasyWheels based on machine vision as described above, and started counting driven distance based on the hall-sensor. When driven distance would have been equal to calculated distance between ReD and EasyWheels, ReD would have turned on destroying motor in the side of centerline, where weed was detected (Figure 35). These distances and marginals, like distance from EasyWheels' weed detection camera to back lights, and marginal needed to turn on and off destroying units before and after weeds, were not measured.

Discussion

It turned out that Colibri's are not the right choice for machine vision computers as they lack a FPU. The missing FPU had to be compensated for some parts by creating algorithms for presenting the lack in mind. For some image processing operations however that was not possible. Colibri's computing power didn't seem to be sufficient for complex image processing. As a navigation computer eBox was a good choice. eBox could run all the navigation algorithms and other secondary processing (networking, parameter management etc.) without any problem.

Windows CE came out only passable. Lots and lots of unnecessary work was caused by Windows CE. Many of its features are poorly documented and many things that a C++ programmer could expect for an operating system to have, were missing. Windows CE provides support for .NET Compact Framework C# version only, no .NET C++. With C# many things probably would have worked better, but no one in the team had previous experience in developing software in C#. Generally .NET as a real time language is a little questionable, although this argument has no proven basis. One choice would be to use a Linux based operating system which come in various flavours, also in real time versions.

There was a lot to do to build the robot and make it to work. The project had repeating problems to keep in schedule. The chassis and axel modules were completed not earlier than 1.5 weeks before leaving to competition and there was too little time to test the robot algorithms on field. Without the simulator it may not have worked at all.

Conclusions

Even though EasyWheels used a novel design, it didn't perform as well as planned. The design is excellent, but due to lack of work and testing time only a small part of the robots potential was used. Modular design definitely was a great improvement to the previous robots; every part (mechanical, electrical or software) could be separately tested.

The amount of work to create this kind of robot is huge. To create a robot like EasyWheels it definitely needs motivated members. It is not possible to master all the areas so it is a must to have people who design the mechanics and people who program and develop algorithms. A year for 6 member team would be barely enough to build EasyWheels, we had 6 members only for 5 months and 4 members for 5 months. EasyWheels was built entirely afresh, but with following robots that is not necessarily reasonable. Nearly every part of the lower level C++ code is reusable with minor modifications, so it would be wise to benefit from that.

Acknowledgments

Building of the robots and participation to the competition in Netherlands was possible thanks to our sponsors: Valtra, Koneviesti, Laserle, HP Finland, HP InfoTech and Suomen Kulttuurirahasto.

References

Backman, J., Hyyti, H., Kalmari, J., Kinnari, J., Hakala, A., Poutiainen, V., Tamminen, P., Väätäinen, H., Oksanen, T., Kostamo, J. and Tiusanen, J. 2008. 4M – Mean Maize Maze Machine. Proceedings of the 6th Field Robot Event 2008. ISBN 978-3-00-027341-4. pp. 9-41. http://www.fieldrobot.nl/downloads/Proceedings_FRE2008.pdf

Honkanen, M., Kannas, K., Suna, H., Syvänne, J., Oksanen, T., Gröhn, H., Hakojärvi, M., Kyrö, A., Selinheimo, M., Säteri, J. and Tiusanen, J. 2005. The development of an Autonomous Robot for Outdoor Conditions. Proceedings of the 3rd Field Robot Event 2005. ISBN 90-6754-969-X. pp. 73-90. http://www.fieldrobot.nl/downloads/Proceedings_FRE2005.pdf

Maksimow, T., Hölttä, J., Junkkala, J., Koskela, P., Lämsä, E.J., Posio, M., Oksanen, T. and Tiusanen, J. 2007. Wheels of Corntune. Proceedings of the 5th Field Robot Event 2005. pp. 75-87. http://www.fieldrobot.nl/downloads/Proceedings_FRE2007.pdf

Telama, M., Turtiainen, J., Viinanen, P., Kostamo, J., Mussalo, V., Virtanen, T., Oksanen, T. and Tiusanen, J. 2006. DEMETER – Autonomous Field Robot. Proceedings of the 4th Field Robot Event 2006, ISBN 978-90-8585-480-7. pp. 27-38. http://www.fieldrobot.nl/downloads/Proceedings_FRE2006.pdf

BaneBots Robot Parts, http://www.banebots.com

CMUcam3, http://www.cmucam.org/

EmbeddedPC.NET, http://www.embeddedpc.net

FlowCode, http://www.matrixmultimedia.com/flowcode.php

Futurlec, http://www.futurlec.com

ICOP, http://www.icoptech.com

Laserle Oy, http://www.laserle.fi

MSDN, Microsoft Developer Network, http://msdn.microsoft.com

OpenCV, Open Source Computer Vision Library, http://sourceforge.net/projects/opencvlibrary/

Toradex, http://www.toradex.com

Eduro – Navigation in the maize

Martin Dlouhý[1,2)], Jiří Iša[2)], Jan Roubíček[3)], Tomáš Roubíček[3)]

1) *robotika.cz*
2) *Faculty of Mathematics and Physics, Charles University in Prague, Czech republic*
3) *www.robsys.cz*

Abstract

This article describes a robot used by the Eduro team for the Field Robot Event 2009. The goal of the team is to build and program a robot with a basic functionality and to gather the data and experience for the next year competition.[1]

Introduction

The Eduro team is a group of people interested in robotics formed across research and industry. We are bond together by a belief, that robots can and should be used for tasks humans cannot perform well or even at all. Our past experience with the robotic constructions and our previous participation on the robotic competitions (Eurobot, Robotour, RobotChallenge, ...) has shown us an applicability of our approach and of our research ideas to the robotics.

The past has also learned us that it is not a lack of ideas which stops the robots from performing well, but a lack of reliability. Because of this observation, we focus on the reliability instead of the complexity. Our approach is further reinforced by a necessary trade-off between computing power of the control board and its power consumption. There are many nice and important research ideas, which just cannot be implemented on a real robot yet. However, some can.

Hardware

Figure 47: RobSys Explorer

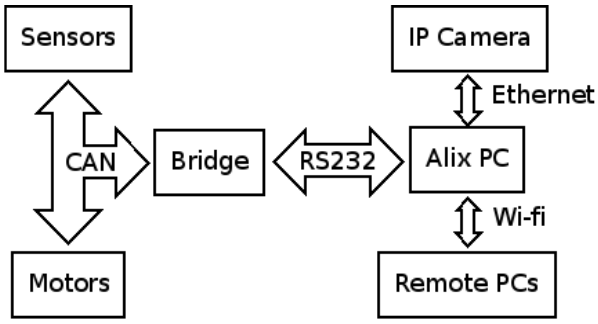

Figure 48: Hardware structure of the Explorer

The robot for FRE is derived from the RobSys Explorer which was originally designed as a remotely controlled inspection system. The base is formed by a rigid water-proof chassis with four wheels (Fig. 1). The robot weights about 9 kg and has the dimensions of 300x280x200mm . Every wheel has an own motor and a controller. The motor controllers and other units are connected via the CAN

1 This research was partially supported by the Program "Information Society", project 1ET100300517.

bus (Fig. 2). The CAN network is connected via a CAN-to-RS232 bridge to an Alix single board computer (AMD GeodeCPU running on 500MHz, 256 MB RAM, compact flash card, wi-fi and ethernet). Vision is provided by a modified D-link DCS 900 IP camera which is attached to the Alix board. Besides the camera, infrared Sharp distance sensors are equipped for an obstacle detection.

Software

The main control program and image processing run on a single board computer (Alix) with Linux.

The low-level motor control runs on the microcontrollers on the CAN-connected modules close to the hardware.

Having a goal of gathering data and experience this year, we are going to experiment with several control algorithms:

1. Monte Carlo localization
2. Insect-inspired navigation
3. Vision-based path recognition

Monte Carlo localization provides a method how to merge readings from different (noisy) sensors into a knowledge about a state of the robot. We have used it successfully in the past, but in the basic implementation it requires a rough map of its environment. We plan to test some ideas about a simplified map and state of the robot description.

Insect navigates through an environment very similar to the one provided by the Field Robot Event, yet it needs a very small computation power. We intend to experiment with an approach based on the optical flow.

For the Robotour competition (http://robotika.cz/competitions/robotour2009), we have designed and implemented a simple, yet robust, vision-based road recognition. According to our preliminary experiments it should perform well in the environment of the maize rows too.

Conclusions

This year, the Eduro team is an experimental participant of the Field Robot Event. We are going to put the approaches we have developed in the past and specially for the Event and our robot under a stress of an environment even rougher than we have dealt with before. The data obtained and experience gained should hopefully be used in the next year competition and in other robotic projects.

Acknowledgments

We would like to thank Zbyněk Winkler and Ondřej Luks for helping us to implement a message processing layer of the control software.

References

http://robotika.cz/

http://www.mff.cuni.cz

http://robsys.cz

http://www.eurobot.org

http://www.robotchallenge.at

http://robotika.cz/competitions/robotour2009/

http://www.can-cia.org

Dellaert, F., Fox, D., Burgard, W. and Thrun, S. (1999). "Monte Carlo Localization for Mobile Robots". IEEE International Conference on Robotics and Automation (ICRA), 1999

Fox, D., Burgard, W., Dellaert, F. and Thrun, S. (1999). "Monte Carlo Localization: Efficient Position Estimation for Mobile Robots". Proc. of the Sixteenth National Conference on Artificial Intelligence (AAAI'99)

Low, D. T. and Wyeth, G. F. (2005). "Obstacle detection using optical flow." Proceedings of the Australasian Conference on Robotics and Automation (ACRA 2005), December 5-7, 2005, Sydney, Australia

Braillon, Ch., Pradalier, C., Crowley, J. L. and Laugier, Ch. (2006). "Real-time moving obstacle detection using optical flow models." Intelligent Vehicles Symposium 2006, June 13-15, 2006, Tokyo, Japan

Eyesonic EVOLUTION

From left to right:

Kees Kroes; Teus van Laar; Jacob-Jan Boonzaayer; Dré Jongmans and Johan Booij.

Contact: info@eyesonic.nl

Abstract

The following report is a description of the field robot EyeSonic EVOLUTION. This robot is an improved version of last years EyeSonic and is build to participate in the Field Robot Event 2009 in Wageningen. A team of five students Agricultural and Bioresource Engineering have been working on the robot for two study periods. The robot is build to perform three specific tasks. It is designed to navigate through rows of corn autonomously. The challenges it faces are curved rows and a curved headland. Therefore the robot is equipped with a laser scanner (Sick LMS 111) for sight and programmed in such a way that it can interpret the data from the laser scanner and use this to find its way through the rows. Besides navigation, the robot also has to be able to find and spray green golf balls that are randomly distributed through the rows of corn. Recognition of the golf ball is done by image processing in Labview. In building and testing the robot, the navigation within the row proved to be a very robust system. The headland turning remained a big challenge up until the last moment. This was mainly due to difficulties in analyzing and processing the changing circumstances on the headland. At the Field Robot Event it turned out most of the programming worked correctly and the overall performance of the robot was quite good. The biggest failure we experienced were a couple of lost screws which resulted in a dangling motherboard. These technical errors are difficult to rule out.

Introduction

In 2009 the Field Robot Event was held in Wageningen again. This is already the seventh edition of the event. Each year in the third study period there is an information afternoon for students who are interested in participating in this event. After visiting this afternoon four of our team members started the Field Robot Project. This project runs trough the fourth and fifth period. At the start of the last period our team was completed with the fifth team member.

The recommendations of the previous robot team told us that we should continue using the current robot, The Eyesonic, as a basis. So we took the old robot and used this as our basis. The robot has been improved on several points. We bought some new hardware parts like a second RoboteQ controller and got a laser scanner. Furthermore we did a lot of work on improving the existing LabVIEW software. This lead to a more accurate and robust navigation.

The team always worked in groups of at least two persons to a certain part of the work. This was done to prevent mistakes and kept creativity alive.

The following challenges where added since last years competition: 1 Golf ball detection with green balls instead of yellow balls, 2 spraying these green golf balls, 3 Curved headlands on the Advanced navigation part and 4 Programming the laser scanner.

Programming the laser scanner was a challenge to us, as we didn't have any examples how to implement this in Labview. The students of previous years used ultrasonic sensors, so the laser scanner was new too us.

Hardware

Figure 1 represents all hardware and how the hardware is connected to eachother. In this paragraph we discuss our mean changes comparing to last years hardware.

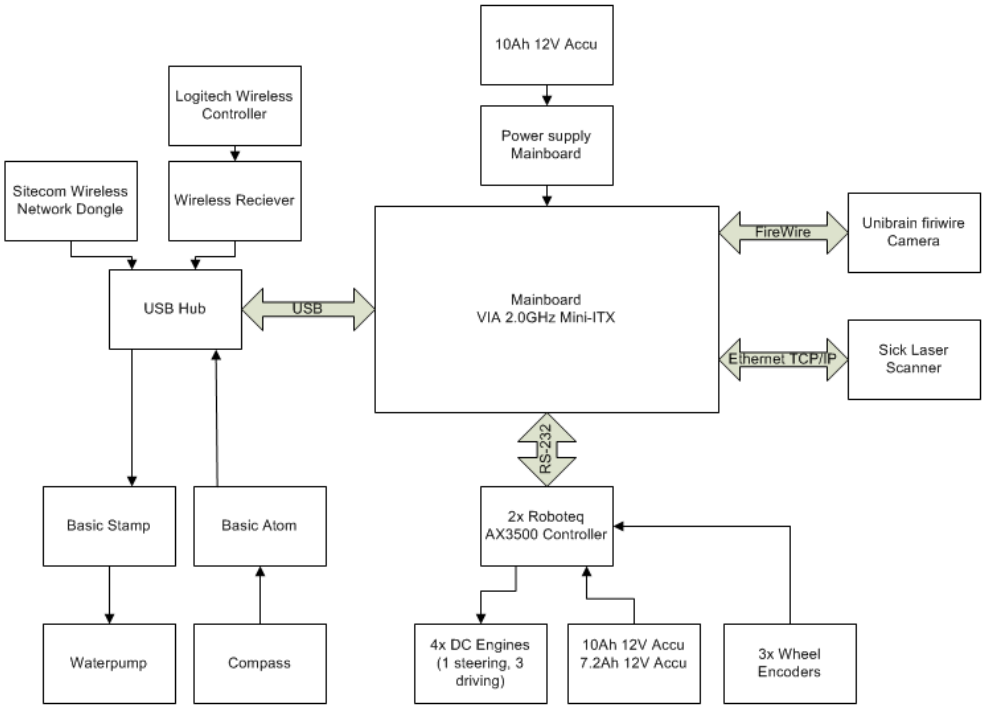

Figure 38: Hardware connectivity diagram.

Chassis for Eyesonic Evolution

The chassis of the robot is the same as is used in 2008. We decided to use this chassis again, because this has many advantages. The box, in which al the hardware is mounted had enough space left for the things we wanted to add. The chassis is very robust and can be used again without problems. Another important reason why we decided to use the chassis again is that it will cost a lot of time to develop a good alternative. One of the problems of the previous team was a lack of time to make the robot more intelligent, because they put a lot of time in developing and constructing the new chassis.

For al this reasons we decided to use the old chassis and just try to improve it on critical points.

Roboteq AX3500

This year we bought and installed a second Roboteq controller. This is of the type: AX3500.

With the combination of this new RoboteQ controller and the old one, we are now able to control the speed of each wheel independently. This takes care of the steering problems. Last years robot tended to push the front wheel forward even if it was in a steered position. This was because the back wheels didn't correct their speed for making a turn. When turning, the outer wheel should turn faster than the inside wheel because it has to cover a larger distance. To find the correct velocity for each wheel, a kinematic model of the robot is made.

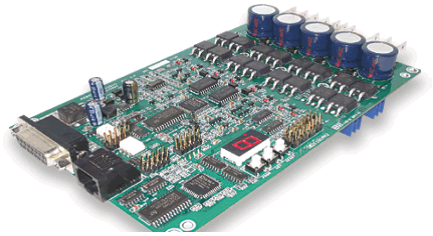

Figure 50: AX3500 Roboteq controller

A roboteq controller has two channels. First we had just one controller, so two channels. One channel was dedicated for all wheel motors and the other channel for steering. Now we have two controllers, so four channels. Each wheel motor is connected to a channel and the last channel is used for steering. This makes it possible to give each wheel motor a separate velocity.

Encoders (closed loop)

Every RoboteQ has two digital encoder inputs so we have four of these. Now we have one input for the speed of each wheel and one for steering. When a wheel experiences more resistance the RoboteQ controller will automatically apply more power to this wheel to keep it at the desired speed.

Sick Sensor

For navigation we used the laser scanner which we received from Sick BV. It is the LMS 111 model.

Figure 51: Sick Sensor LMS111

We tried to make a proper working and robust program for navigation between maize rows. This scanner was fitted to the front of the chassis. The scanner based its measurements on distances to the rows of corn. To minimize leaf disturbance we made a bracket in such a way that we could easily adjust the height at which the sensor is scanning. This can be adapted to the crop conditions. When corn has the size it has at the Field Robot event it tends to have many leaves that obstruct the view of the row. Closer to the ground there are less of these leaves and the row gives a better measurement. The scanner therefore was mounted with the mirror on the bottom. So upside down from the picture in figure 9. We couldn't go much lower because then the sensor mounting would hit the front wheel and disturb the steering.

Logitech Controller

We installed an usb receiver on the computer. The receiver communicates with a Logitech rumblepad wireless controller. With this device we can control the robot functions: steering, throttle, emergency stop, and the selecting of drive mode.

We have the following drive modes:

- Hand control (manually steering of the robot)
- Navigation (autonomously navigating through the field)
- Weed Control (autonomously navigating and spraying golf balls)
- Freestyle (autonomously navigating, close to the right-hand row)

Spraying hardware

For spraying we use a pump and water reservoir from a car window washing system. This system can spray the front and rear window of a car separately by pumping in two different directions. We use this feature to spray left or right separately by switching the polarity of the current.

To control this motor we need a relay or controller which has three outputs: off, on and change poles. We use a NV2R relay for chancing poles and we use an ULN803A chip to control this relay. To control the chip we have a basic stamp. Figure 4 shows the wire connection of this part.

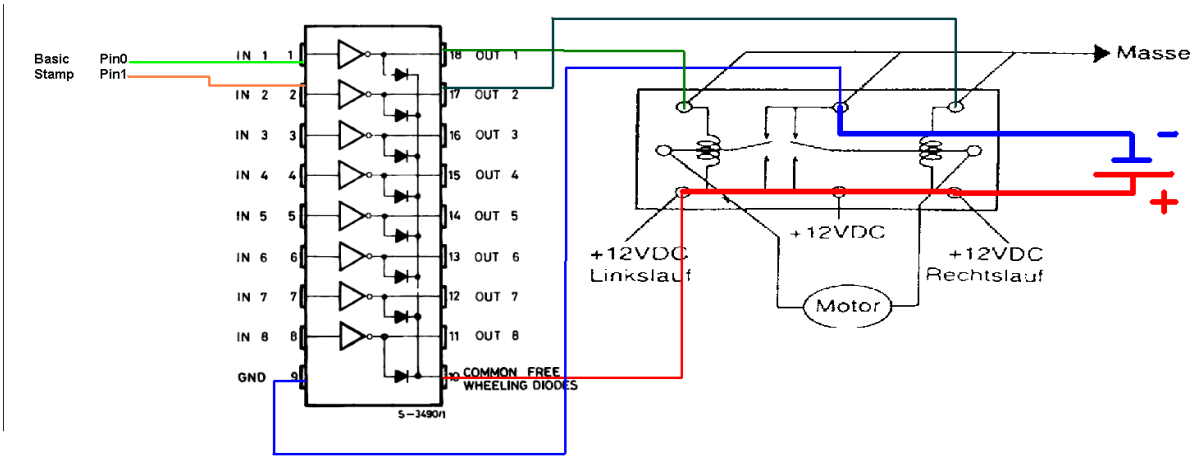

Figure 41: Wire connection of the basic stamp for controlling the sprayer

Connected to the frame of the robot we made two simple spraying arms of aluminum. These are only 5 cm wider than the frame of the robot to keep it from damaging the crop. In these spraying arms we drilled a hole that exactly fits a spray nozzle. This is attached directly to the pump through a tube that runs outside the robot frame. The spray nozzles we eventually chose are used in agricultural sprayers. We chose these because they give a conical spraying image. This gave the nozzle a wide surface perpendicular to the robot but narrow linear to the robot. In this way it is possible to hit the ball with relative precision.

Software

Almost all the software is written in the graphical programming language Labview 8.5. The operating system installed on the system is Windows XP professional.

The program consists of three modes for the different tasks and there is also a mode for letting the robot stop driving. These modes can be chosen with the Logitech controller.

In figure 4 we can see all the modes and there underlying software components.

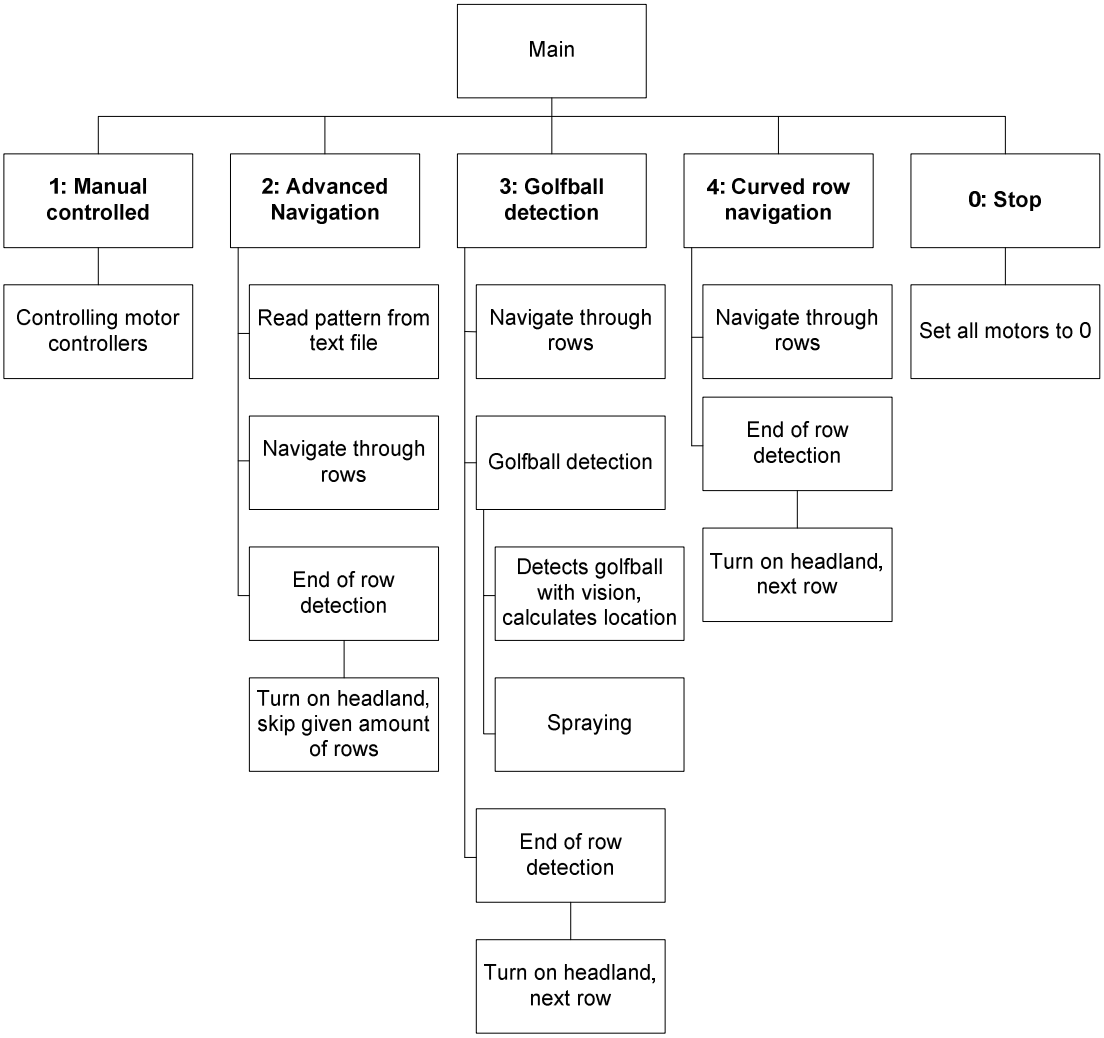

Figure 42: Overview of the software components.

Hand Control Mode

For hand control mode, the '1' button on the Logitech controller is used. The program uses then only the information (velocity and steering angle) which comes from the analog joysticks of the Logitech. This information is used by a Virtual Instrument (VI) that manages the motor controllers.

Advanced Navigation

In Advanced navigation modus, the program has to do much more. There are roughly three different tasks to do:

- Navigation within the rows: driving.

- Navigation within the rows: End of row detection.

- Headland navigation.

For all these tasks, the velocity is given by a constant.

Navigation within the rows: driving

For navigation we used the laser scanner which we received from Sick BV. We tried to make a proper working and robust program for navigation between maize rows.

In previous years ultrasound sensors were used. The navigation on these sensors was based on the difference between the distance measured on the left side, and the distance on the right side. This principle is also applicable when using a laser scanner. Three intervals on each side of the robot are selected, next the smallest measured distance of each interval is used. So for every interval one point with an x,y coordinate will be used. With these six points a calculation is made for the correction which is needed for the steering angle. In figure 5 are the six points visible.

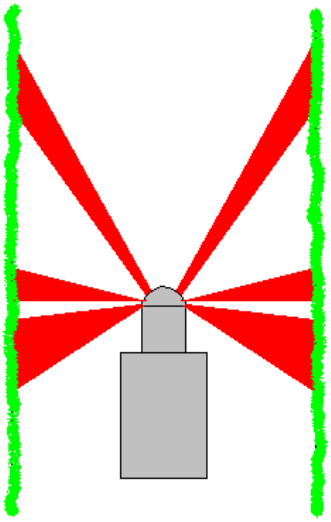

Figure 54: Sick sensor looking to both rows

In figure 6 are the measurements from the laser scanner plotted. The maize rows are clearly visible.

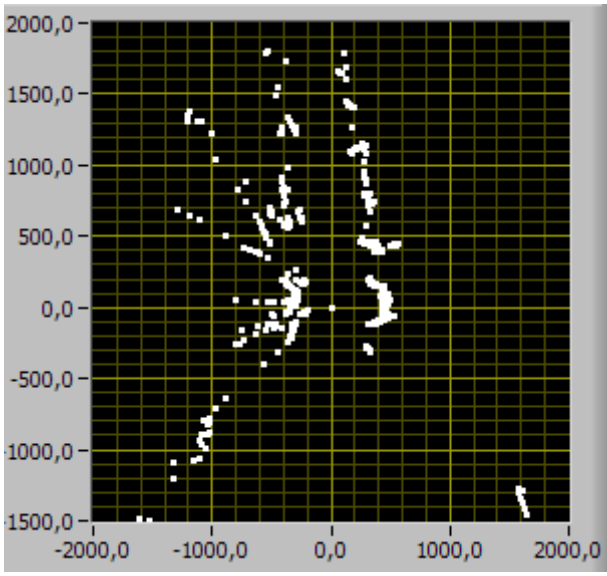

Figure 55: x,y plot of the measurements with the sick sensor.

In Labview, we made a case structure with possibilities to navigate on one maize row, or on just a few points (two or one) on each side.

In table 1 can be seen how the robot reacts on each situation.

Now the navigation seems to be more robust. Field tests affirmed this: the robot is now able to deal with gaps in the maize rows.

Table 1: case structure

Situation	If sensor receives…	Solution
normal row	distance less than 60 cm at both sides	drive on information of all intervals
gap in one row	distance of more than 60 cm at one side	drive on information of the other row
gap in both rows	distance of more than 60 cm at both sides	drive on information of the intervals less than 60 cm, otherwise: go straight on
headland (both sides)	distance of more than 160 cm at both sides	go to headland modus
curved headland	distance of more than 60 cm at destinated side, and 160 cm at the other side	go to headland modus

Headland turning

When the laser scanner gets no values beneath 160 cm on one side and no values beneath 60 cm on the other side, the robot concludes it is on the headland. This difference between both sides made it able to deal with a non perpendicular headland. Also an area of 160 cm in front of the robot should be empty.

The robot starts turning when the minimum distance at the side to which the robot has to go, is above 60 cm. Then the robot makes a turn until it sees the maize row within 60 cm. This is measured by the compass. Then the robot navigates perpendicular to the rows by keeping a distance of 60 cm to the rows. For distance we read the average distance of the two back wheels by reading the wheel encoders. When the predefined distance is traveled, the robot makes another turn to complete the turn of 180 degrees compared to the position were it came out of the row. Then the program for navigation in the row will take over the work again, and the robot goes its way.

Weed Control Mode

For weed control the same program is used as is used for navigation. The velocity is roughly half of the speed for navigation tasks. (1 m/s). The camera looks for golf balls, and when one is detected, it calculates the distance until spraying, makes the velocity temporarily lower(for more accuracy), sprays the ball, returns to normal speed and keeps on navigating. Headland detection is the same as used for navigation.

For detecting golf balls we filter the balls out of the incoming images on color and shape.

First we use a model to extract the red, green and blue planes from the images and process them to a 8-bit image. This process is realised by adding the green and blue plane to each other and subtracts the red plane. We use the blue plane because with our camera settings we found that the “green” golf balls emitted blue light when the light intensity went up.

Second the image is transformed to a binary image, which makes further processing possible.

Third all small particles are removed and the remaining particles are closed to get a more filled circle for better circularity detection. After that we used the Heywood Circularity Factor to look for complete round balls and an area filter was applied to get the right amount of detected golf balls.

In figure 7 the result is visible.

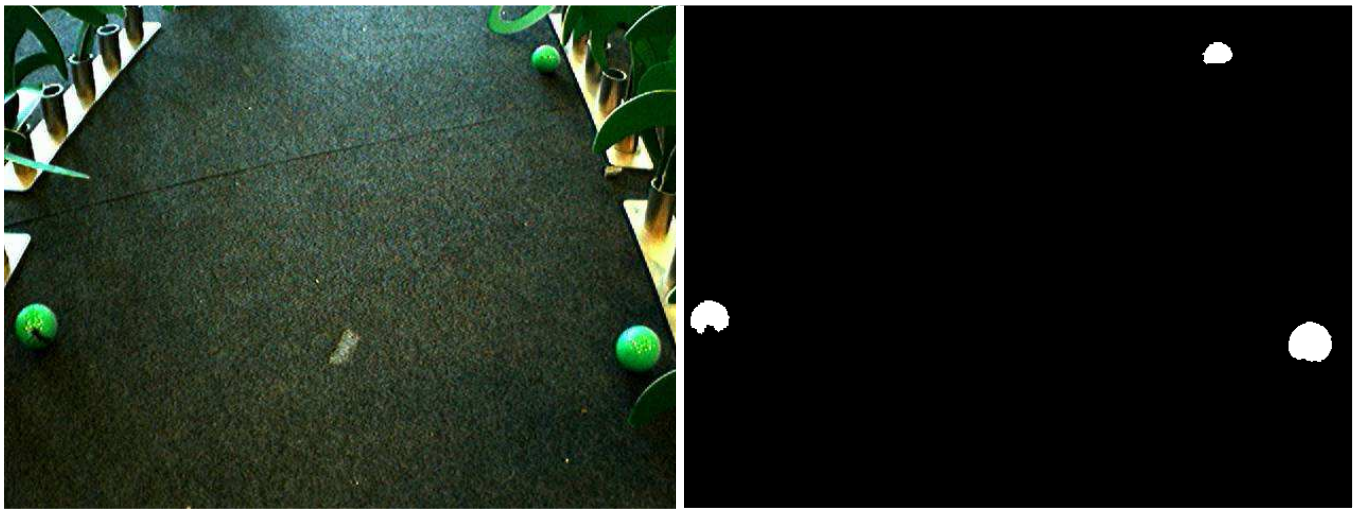

Figure 56: Left is the incoming image, right is the processed image

To complete the program, the images are calibrated and analysed to know the distance between the golf balls and the robot. The program gives a signal to the basic stamp, which process this signal further too a relay. This relay controls the pump for spraying and the action will be performed.

Weed Control

This task was build up inside a tent, due to the bad weather. The crew build up two straight maize rows of fake plants with green golf balls between it. Cause this task was held inside a tent, the light conditions were very poor. Our camera didn't see all the balls correctly. That's why the robot detected two out of fourteen golf balls. But these golf balls were sprayed very accurate comparing to other robots. This and the fact that our robot drived very smoothly and in a straight line due to the sick sensor , resulted in a high grade of the jury. We got the first place in this task.

Navigation

In this task the robot had to drive through curved rows of maize. The robot has to cover as much distance in three minutes as possible. A good point of our robot is that it's very fast comparing to other robots. It drove one meter per second through the field. A bad point was that we had some technical problems with the computers mainframe fitting, which resulted in touches to correct the robot. This task we finished fourth.

Advanced Navigation

In this taks the robot should follow a patern through the maize field. Our technical problems were roughly solved, so the robot started this task. The robot raced through the maize field and followed a certain pattern. There were some problems at the headturns, which resulted in some touches. When we looked at the race, we thought that our robot did very bad comparing to other robots, looking at the touches also. But afterwards we heared that our robot covered the biggest distance at this task and we went home with the first price voor Advanced Navigation.

Free style

In this task the robot should do something, which has a agricultural background. We had to think out something ourselves. First we wouldn't participate in this task, as we focussed us on the other tasks.

However at the Event one of our group member had a genious idea. We attouched a Stanley-knive to the robot and we drove the robot manually through the maize field, whereas the knive hitted the plants. This was the way we transformed our robot to a mechanical maize harvester.
Supporters liked this idea, as it was improvised in this short time, it worked pretty good and showed a simple and effective alternative use for the robot. In this task we got the fifth place.

Over all

The point total from the jury decisions for the different parts off the competition combined with the points given for technical performance resulted in a first place overall.

Discussion

At the start of the project we used the EyeSonic as a base for our own robot. There were a number of basic elements of this robot we could use very well. The closed chassis and the computer were updated last year so these still worked up to standards. We also looked at the recommendations of last year's team to see what could be improved on this robot. Also general problems were identified. We would like to do the same thing.

To start with the basic robot (the chassis and drive train), there are already some points that need improvement. Last year's team build a aluminum chassis where all components could be brought inside. This frame is large enough to fit all components and makes the robot robust and able to drive

outside in less than perfect conditions (e.g. rain etc.). The three wheel setup worked very well for us. It enables accurate and close quarter steering. We therefore recommend to continue using this setup as a frame and steering mechanism. However not all components of the chassis are up to standards. The wheel motors we used this year are ready to be replaced. The axels are bent which results in a distortion of driving smoothness. In practice this results in a "wobbling" motion of the robot.

Adding to this is the lack of suspension. Almost all other robots competing in the Field Robot Event have suspension. At the event we found that the robot driving at 80% of its power is still able to navigate and drive properly although it jumps around a lot. The shaking of the chassis did result in the loss of nuts and bolts that are attached the motherboard of the computer. This resulted in the computer getting stuck during task one which cost us a lot of meters in this part of the competition. To improve driving smoothness suspension on all three wheels could prevent these errors in the future. This will lead also to a much more consistent and smooth data collection.

Furthermore if you are looking for new and possibly more powerful motors it is also a good idea to look at the power supply. We already bought 3 new lead acid batteries.

Second the headland turning kept giving us challenges. One of the biggest problems with headland turning is keeping track of your location when you don't see rows of crop anymore. We wanted to use a combination of the compass and the Sick sensor to navigate. The compass isn't fail proof because of it's sensitivity to moving metal and magnetic parts. Therefore it would be a good idea to look for another means of position determination. At the event we fell back on the old fashioned navigation based on a given time and given travel distance. This works but isn't a very robust solution. A little amount of slip on the wheels already disturbs the headland turning.

Third the golf ball detection could be more robust and accurate, if the settings of incoming images are every time the same (brightness, contrast, gamma, etc.)

Helios had a nice solution for this problem. They used a cut in two green golf ball, which are pasted to the front of the robot, as a reference point. In this way the robot knows the exact color of the green golf balls for compensation.

And then finally we recommend to look at robustness when building a chassis. This years weather conditions once more showed that Field Robots should be able to withstand rain and other difficult conditions. It's a good thing to keep this in mind when developing a robot.

Conclusions

Competing in the 2009 Field Robot Event brought us the first price for the Advanced Navigation and the Weed Spraying challenge and a first place overall. This was more than we expected starting at the competition. The good result was partly thanks to this year's new sponsor of our team Sick bv. By supplying us with their laser scanner. This made a significant improvement in navigation possible. With this faster means of navigating we were able to utilize the full capacity of the chassis and driving motors of our robot. The sponsoring of Lely helped us further by making it possible to buy some additional components which resulted in entering the competition with a robot which had all the potential to win the contest. A technical error resulted in a couple of problems during the contest. The computer getting stuck was besides the failure of the compass the only problem we encountered. This led us to the conclusion that our hard- and software were correctly developed for this competition. Further precision in golf ball detection and headland turning is most definitely possible but the way for normal navigation is proven to be pretty full proof.

Acknowledgments

The Eyesonic Evolution team would like to thank some people, without there help the task of building a robot would by very difficult. Firstly we would like to thank the Agrotechnology student teams of previous years, for their help, support and input at difficult times. Secondly we would like to thank Sam Blaauw, Ard Nieuwenhuizen,Kees van Asselt and Tijmen Bakker for their help, technical know-how and support in difficult times. Thirdly our thanks goes to the FTE and SCO groups of Wageningen University for given us as students the change to participate in this contest. Fourthly we would like to thank the organization and crew of the Field Robot Event 2007. Also we would like to thank Jan Willem Hofstee for his help on Machine Vision and his support during the whole development process.

Every year the student team works hard to deliver a robot that performs as optimal as possible. In order to equip it with the best technology they can find for the tasks they rely on sponsors for financial support. Without the help of the following sponsors the realization of the EyeSonic Evolution wouldn't have been possible.

Development Work Shop AFSG Wageningen UR

Sick Nederland BV

Lely Industries

References

www.maxonmotor.com, www.fieldrobotevent.nl, www.eyesonic.nl, www.sick.nl, www.basicmicro.com

(2000). "Quick Assembly Two and Three Channel Optical Encoders." Technical Documentation Maxon Motor Control.

N.D. Tillet et al. "Inter-row vision guidance for mechanical weed control in sugar beet". Computers and Electronics in Agriculture 33 (2002) 163-177.

(2000). Basic Stamp manual version 2.0b, www.parallaxinc.com.

(2003). Introduction to Labview, Six-Hour course. National Instrument Corporation, www.ni.com, September 2003.

(2005). AX3500 Motor Controller User's Manual. v1.7, Februari 1, 2005. www.roboteq.com

J.P. Feddema et al. "Proceedings Field Robot: Eyesonic". June 2008, Wageningen, The Netherlands.

Project M.T.R. – The Mobile Terrain Robot

J. van de Mortel
j.vandemortel@student.fontys.nl

Abstract
Project M.T.R. was a robot built by one person for his hobby. The platform was then used for the Field Robot Event 2009 at Wageningen. The main purpose is to navigate through a maize field autonomously. The robot had wireless communication so it could be controlled over a distance, but since we used vision it got his own laptop. The laptop analyzes the vision data and sends commands to the robot. We still have to do a lot of work, this is because we started the project late april 2009. So we didn't have much time for it, we also had to do it in our spare time. The idea was to gain some experience for the next Field Robot Event.

1. Introduction
We started this project with a group of 4 students of the Fontys University of Applied Sciences Venlo second year. We started late april 2009 with a very low budget. Because the robot platform already

was made by one student. The only thing we had to do is make the control software and add some sensors. The control software was made with Labview 8.6 and Labview Vision. After testing a lot of vision algorithms we chose one. Because of the late starting date we still have a lot to do. But it is a nice learning process that we can use for the next Field robot event.

2. Hardware

2.1. Robot Platform

The platform was build in 2007, 2008 for hobby purposes, the idea was to keep it simple and cheap. The platform is made from a 3 mm aluminium sheet that is bended to a U shape. The robot had 6 wheels the wheels on the side are connected together with a bicycle chain. The wheels are driven by a 200watt DC motor that is connected to the axle by some gears. The platform is build for approximately 200 euro's. The main problem is that it is to big for a maize field it just fits, because it wasn't originally built for the Field Robot Event.

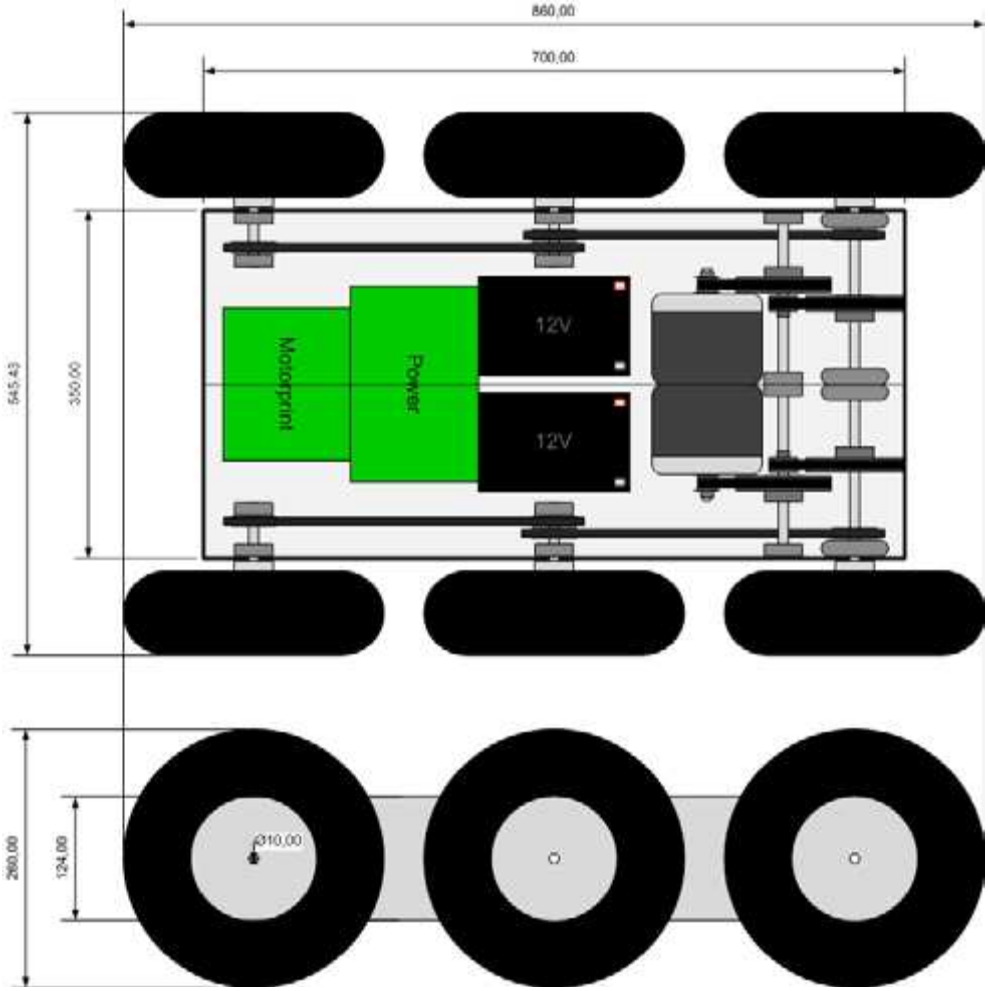

Figure 2.1.1: First drawings robot Platform.

2.2. Electronic's

The electronic's are all self designed. The robot has a mainboard with a microcontroller PIC18F8723 from microchip. The microcontroller communicates with the PC by RS-232, and communicates with the sensors that are on the robot. The mainboard receives data from the PC and controls the motorboard. The motorboard has had dual H-bridge that is capable of delivering 1500 Watts of continues power at 24V. The mainboard and motorboard are optically isolated. The robot is powered

by 2, 12V lead-acid battery that are connected in series to create a 24V power supply. All the PCB's are designed with EAGLE 4.11.

2.3. Sensors
The robot has several sensors, but only one used. That is de digital compass that can be read by a I2C. The robot can also measure the battery voltage.

2.4. Vision
The robots main "sensor" is vision. The robot navigates based on the images that are coming of a Webcam. At first we used a CCD camera module, but the problem was that the color depth was very poor. So three days for the event we found a webcam, the Logitech Quickcam Pro 9000. The image quality is so much better.

3. Software

3.1. Electronic software
The software for the microcontroller was written in C with the Microchip MPLAB C18 compiler. The software isn't complex. The only thing it does is read de compass sensor read the battery voltage update the LCD and controlling the H-bridge. The software receives and sends data to the PC.

3.2. Vision software
The robot's main software is made with Labview 8.6 and Labview Vision. What the software does it analyses the image retrieves a angle of the robot and the rows, with this angle the robot corrects it selves and drives forward. The algorithm that is used to retrieve a angle is called "Visual Sonar". What it does draws lines from the bottom to the top until it sees an object. Then you get an array with all the lengths of the lines. With this you can draw a sonar image (see figure 3.2.1. the white part).

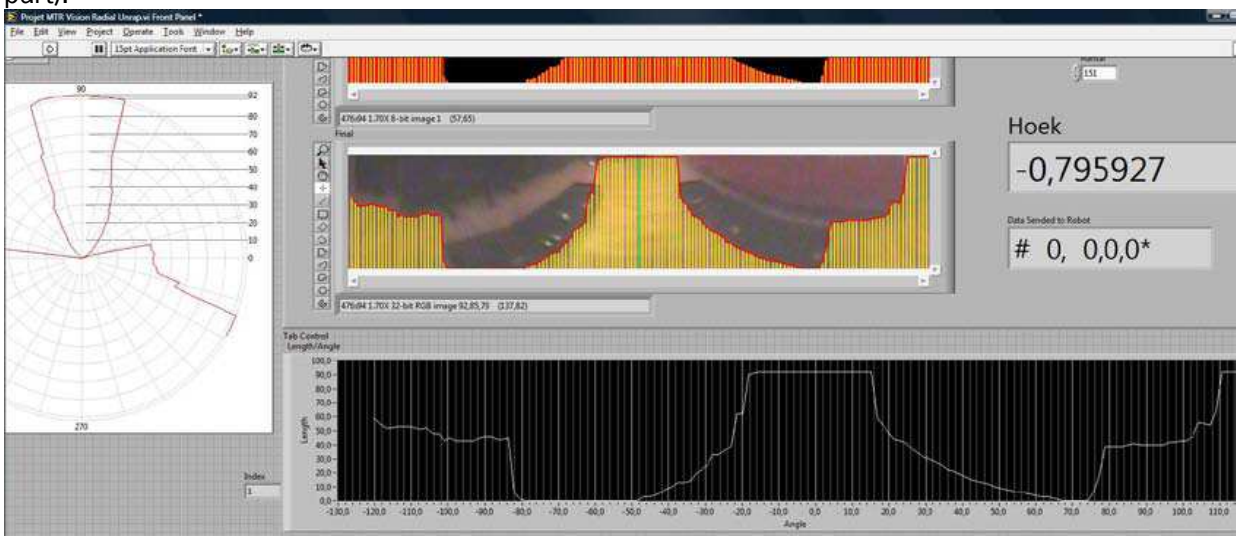

Figure 3.2.1: Labview Vision with Visual Sonar.

3.3. Weed detection
We used a simple algorithm to detect the golf balls, we color threshold the image in the HUE plane. Than we search the picture for a cluster of pixels with a minimum size and a maximum. If the cluster of pixels is between the minimum and maximum we see it as a ball, and we count it. The weed spraying action we didn't do because we hadn't enough time for it.

4. Conclusion

After three months of work we came really far with our knowledge and budget. Although we have a lot to do it was a great learning experience. We know for the next time we have to build a smaller robot, because this one is to big so there isn't much room for steering error. The vision system is a little bit slow mainly because of the drivers in Labview. And we can only use one USB camera with Labview. We also need to use more sensors so the robot can navigate better through the maize field.

References

Boyan Bonev, Miguel Cazorla and Francisco Escolano. " Robot Navigation Behaviors based on Omnidirectional Vision and Information Theory".